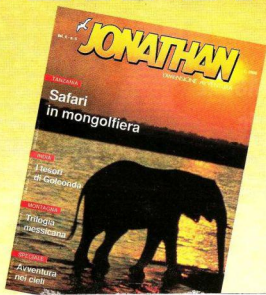
Anno IX - N. 79 - 25 NOVEMBRE 1990 - Sped. Abb. Post. Gr III/70 - CR - Distr.: Parrini
Commodore
COMPUTER
CLUB
#79
L. 6.000
La rivista degli utenti di sistemi Commodore
AMIGA
Qui parla il modem
I basic compilati
Grafica animata
C64/128
Il 6499 è al telefono
Caro file, ti scrivo
"C" la danza dei cerchi
MS-DOS & TURBO PASCAL
Partiamo alla grande
Systems

DIMENSIONE
AVVENTURA
JONATHAN
OGNI MESE
IN EDICOLA
JONATHAN
DIMENSIONE AVVENTURA
TANZANIA
Safari
in mongolfiera
INDIA
I tesori
di Golconda
MONTAGNA
Trilogia
messicana
SPECIALE
Avventura
nei cieli

Sommario

Amigames

Da pagina **43** a pagina **49** la consueta rassegna dei videogames in arrivo; ed i relativi commenti, severi come al solito!

Campus 64 / 128

Campus Amiga

Usa il tuo computer

Rubriche

E' bene ricordare che...

Parleremo del **C/128** solo fino alla fine del 1990.
Dal **gennaio '91** verrà evasa prevalentemente la corrispondenza pervenuta a mezzo BBS (modem).
Dal **gennaio '91** verranno privilegiate le collaborazioni che perverranno in Redazione a mezzo BBS.
Tra breve verrà dato ampio spazio, oltre che ad **Amiga**, anche al favoloso mondo **Ms-Dos**! (ma già lo stiamo facendo...)
Dal **dicembre '91** non verranno più affrontati argomenti relativi al C/64.
Natale è ormai alle porte...

Commodore Computer Club

COMMODORE COMPUTER CLUB

Direttore: Alessandro de Simone
Coordinatore: Marco Miotti

Redazione / Collaboratori:
Davide Ardizzone - Claudio Baiocchi
Luigi Callegari - Umbero Colapicchioni
Donato De Luca - Carlo D'Ippolito
Valerio Ferri - Michele Maggi
Giancarlo Mariani - Domenico Pavone
Armando Sforzi - Dario Pistella
Fabio Sorgato- Valentino Spataro
Franco Rodella- Stefano Simonelli
Luca Viola

Direzione:
Via Mosè, 22 cap. 20090 OPERA (Mi)

Telefono 02 / 55.50.03.10
Fax 02 / 57.60.30.39
BBS 02 / 52.49.211

Pubblicità:
Leandro Nencioni (dir. vendite)
Via Mose', 22 20090 Opera (Mi)
tel. 02 / 55.50.03.10

Emilia Romagna:
Spazio E
P.zza Roosvelt, 4 cap. 40123 Bologna
Tel. 051 / 23.69.79

Toscana, Marche, Umbria
Mercurio s.r.l. Via Rodari, 9
S. G.nni Valdarno (Ar)
Tel. 055 / 94.74.44

Lazio, Campania
Spazio Nuovo
Via P. Foscari, 70
cap. 00139 Roma
tel. 06 / 81.09.679

Abbonamenti: Liliana Spina
Arretrati e s/w: Lucia Dominoni

Tariffe: Prezzo per copia L. 6000
Abbonamento annuo (11 fascicoli) L. 60000
Estero: L. 100000 - Indirizzare versamenti a:
Systems Editoriale Srl c/c 37952207 oppure
inviare comune assegno bancario non
trasferibile e barrato due volte a:
Systems Editoriale Srl (servizio arretrati)
Via Mose', 22
cap. 20090 OPERA (Mi)

Composizione: Systems Editoriale
La Litografica Srl Busto Arsizio (Va)

Registrazioni: Tribunale di Milano
n. 370 del 2/10/82

Direttore Responsabile: Michele Di Pisa

Spedizioni in abbonamento postale gruppo
III. Pubblicita' inferiore al 70%

Distributore: Parrini - Milano

Periodici Systems: Banca Oggi -
Commodore Club (disco) - Commodore
Computer Club - Commodore Computer
Club (disco, produzione tedesca) - Computer
- Computer disco - Electronic Mass Media
Age - Energy Manager - Hospital
Management - Jonathan - Nursing '90 - PC
Programm (disco) - Personal Computer -
Security - Software Club (cassetta ed.
italiana) - TuttoGatto - Videoteca
VR Videoregistrare

Dal Basic in poi

Quando, alcuni anni fa (ma si tratta, in effetti, di un decennio...) scoppiò il boom dell'informatica, il linguaggio più "parlato" fu, sicuramente, il **Basic**. Ogni computer possedeva una versione del popolare interprete; questo, anzi, era installato stabilmente su **Rom**. Senza tema di smentite possiamo addirittura affermare che il computer girava "attorno" al Basic, e non viceversa.

Oggi le cose son cambiate: c'è la macchina (perfetta, veloce, sicura, espandibile) ed i linguaggi che ad essa si adeguano. E si aggiornano.

Continuare ad operare sempre, e solo, con il Basic, quindi, è oggi un controsenso accettabile solo se consideriamo chi acquista un computer per la prima volta e trova difficoltà, addirittura, a caricare un gioco memorizzato su cassetta... Ma ci sono anche loro, i **principianti**, non dimentichiamolo.

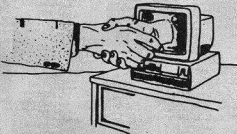
L'esplorazione del **C** (iniziata sul numero scorso), la scoperta del **Turbo Pascal** (quello "buono", però) e l'accettazione del **Quick Basic** (ma solo perchè assomiglia molto al Pascal ed al C, e ne facilita l'approccio) sono accettati di buon grado dall'hobbista evoluto, nostro tradizionale lettore.

Il mondo va sempre più verso il sistema **Ms-Dos**; verso l'**Amiga** (non se ne dispiacciano i patiti del caso) un po' meno. Con il **C/64**, in ogni caso, il discorso è chiuso, pena il rischio di assomigliare agli anziani frequentatori di enoteche che, tra un lambrusco ed un bianchetto, rievocano i tempi andati trattenendo le lacrime.

Da gennaio '91, quindi, la svolta totale della nostra testata; che si muove, e si è sempre mossa, assecondando i desideri dei suoi sostenitori.

Alessandro de Simone

LA VOSTRA POSTA

(a cura di A. de Simone)

Correzione automatica

Esiste un programma in grado di correggere automaticamente gli errori che commettiamo scrivendo un programma in Basic?
(Anonimo del secondo millennio)

Magari! Se esistesse un programma del genere non ci sarebbe più necessità di imparare a programmare. Non è infatti possibile realizzare una procedura del genere, per diversi motivi. Anzitutto il computer non può "sapere" se una determinata elaborazione è corretta oppure no. Facciamo un esempio. Tutti(!) noi sappiamo che l'ipotenusa di un triangolo rettangolo è data dalla seguente relazione matematica...

Ipot. = Sqr (cat1^2 + cat2^2)

Se, invece di scriverla come, appunto, indicato, inserissimo un comando del genere...

Ipot. = Sqr (cat1^2 + cat2)

...dimenticando, cioè, di elevare al quadrato anche il valore del secondo cateto, il computer **non può** accorgersi dell'errore perchè non può sapere che, in quel momento, la formula che abbiamo inserito nel programma si riferisce all'ipotenusa; "lui" elabora la radice quadrata (**sqr**) come una qualunque altra radice.

Un altro esempio, più terra - terra. Supponiamo che, ad un certo punto, vogliamo cancellare lo schermo prima di far apparire un messaggio. Se dimentichiamo il comando specifico *"Cancella schermo"* (presente in qualsiasi linguaggio di programmazione), il computer non potrà mai leggere il nostro pensiero per sapere se, in quel preciso momento, volevamo la cancellazione dello schermo, oppure no. I messaggi che compaiono durante un'elaborazione, pertanto, si riferisco esclusivamente ad errori **formali** commessi dal programmatore: per esempio Pront invece di Print, A$ = Nome invece di A$ = **"Nome"** e così via; del resto, poveretto, l'elaboratore non può (e soprattutto, **non deve**) prendere iniziative.

INiente eccezioni

Diversi utenti insistono nel fare richieste specifiche che poco possono interessare i nostri lettori; oppure nel proporre, ancòra per posta, articoli e programmi per il nostro periodico. Sono costretto a ripetere che non è assolutamente possibile prendere accordi per iscritto, ma solo per telefono. Non me ne vogliano, tra gli altri, *Alessandro Dreani, Raffaele Pescarini* (a proposito, abbiamo pubblicato numerosi progetti hardware, non te ne eri accorto?...), *Renato Pompei, Diego Pozzi, Mauro Campanelli.*

Schede impossibili

Esiste una scheda in grado di trasformare un C/128-D in Amiga?
(Marco Romagnuolo - Portici)

Non penso che possa esistere una scheda del genere, nè ora, nè mai; ad ogni buon conto, prova con il Bancomat...

Pot (n) sul C/128

Con l'istruzione Pot(n) sono risucito a scrivere un programma, in Basic, che simula il funzionamento di un oscilloscopio. Purtroppo il programma è lento e sono costretto a limitarmi a misure di basse frequenze. E' possibile aggirare la lentezza del Basic scrivendo un programma simile, ma in linguaggio macchina?
(Claudio Indino - Civita C.)

Per carità! Non fare altri esperimenti del genere. L'istruzione **Pot(n)** preleva direttamente, dalle porte joy del computer, il valore della resistenza ivi presente. La resistenza di cui parlo è quella dell'eventuale **paddle** (parente lontano del joy) inserita nella presa joystick. Se, invece di una resistenza "passiva" (che, cioè, non presenta tensioni o correnti), il computer trova una corrente elettrica (come quella, appunto, necessariamente esistente in frequenze elettriche, per

Attenti alla SIP

Abbiamo notato, esaminando i collegamenti effettuati con la nostra banca dati, che una gran parte dei nostri utenti effettuano contatti telematici nei pomeriggi di sabato e nell'intero corso delle domeniche. E' probabile che la scelta di queste fasce orarie sia dettata dall'errata convinzione che il costo della tariffa sia dimezzata rispetto all'orario considerato "normale" (ciò accadeva anni or sono) Come, invece, è possibile rilevare (pagina 10 di un qualsiasi elenco telefonico), le fasce orarie a tariffa dimezzata sono limitate al periodo di tempo compreso tra le ore **22:00** e le **8:00** di ogni giorno della settimana, compresi sabato e domenica! La SIP, chissà perchè(?...), non ha diffuso la notizia relativa alle nuove fasce orarie e la gente pensa ancora che l'intera domenica sia disponibile a metà prezzo. Occhio all'orologio, quindi, anche per effettuare comuni telefonate "vocali" in teleselezione.

quanto modeste siano), rischi di combinare un bel pasticcio. In linea di massima è possibile, mediante un adattatore elettronico, evitare guai al computeri.

Rimane, comunque, il problema di far tracciare, in alta risoluzione, ed in tempo reale, la traccia lasciata dal pannello elettronico. Come tu sia riuscito a fare tutto ciò, per giunta operando in Basic, è per me un mistero; sul quale, però, ti sconsiglio di insistere.

Amighista o C/128ista?

Dedicate troppo spazio all'Amiga e ben poco al C/128, computer che posseggo e che mi soddisfa. Ma veniamo al dunque: potete pubblicare la mappa della memoria di Amiga ed una serie di articoli sull'Assembly di questa meravigliosa macchina?

(Amigo di Albano L.le)

Il motivo per cui parliamo più di Amiga che del C/128 lo fornisci tu stesso, magari senza volerlo!

Pur dichiarandoti felice possessore di C/128, infatti, richiedi informazioni su Amiga e non sull'obsoleto computer. Per quanto riguarda Assembly e "mappa" di memoria (ma con Amiga è improprio parlare di mappe, come sei invece abituato con il C/128), ci stiamo attrezzando.

Un po' di pazienza.

IPoke chiarita

Forse ho scoperto il mistero dello strano messaggio che il C/64 emette quando si digita **Poke 22, 0**. Questa locazione, infatti, svolge il compito di puntatore allo stack delle stringhe transienti. Impartendo quel particolare comando, si costringe il puntatore a puntare al messaggio *"Undef'd' functioN Veryfy Error"* (posto in Rom). La **"N"** finale, essendo **shift**ata, compare come una barra inclinata se il set dello schermo è settato come maiuscolo - grafico.

(Massimo Morelli - Cerro M.)

I migliori per C/64

Quali sono i migliori programmi di W/p, data base e spreadsheet per C/64?

(Roberto Biancucci - Nodica)

Easy Script (W/p) e Calc Result (S/s); per il data base dipende dall'uso che ne devi fare. Prendi il primo che ti capita e diventa padrone dei vari comandi disponibili applicandoti seriamente: vedrai che, almeno nei casi più generali, ti troverai bene. Ma non ti pare che, per un uso professionale, il C/64, oggi, sia un po' "stretto"?

C/128isti di tutto il mondo, uniamoci!

Sono un C/128ista irriducibile e, dal momento che tra poco abbandonerete questo computer, vi chiedo di pubblicare il mio indirizzo in modo che, i superstiti di questo computer, possano avere ancora un punto di riferimento per scambi di idee e programmi (in modo, ovviamente, 128).

Enrico Ceppi
Via S. Maria, 14
20036 Meda (Mi)
Tel. 0362 / 72.04.7

Detto, fatto. Dubito, però, che risponderanno in molti. Del C/128 (e del C/64) tutto quello che c'era da dire è stato già detto e, tranne il caso degli ultimi acquirenti del popolare computer, gli utenti non hanno interesse ad acquistare una rivista che, per forza di cose, non può fare altro che ripetere quanto già detto da alcuni anni a questa parte. Credetemi, C/64isti e C/128isti: se vi procurate gli arretrati di Commodore Com-

Copiare Geos

Visto che il Geos obbliga ad utilizzare i preziosi dischi originali ogni qualvolta lo si voglia usare, ecco un sistema per ottenerne una copia, anche se con qualche limitazione. I problemi che si incontrano tentando di effettuare copie di Geos sono due. Il primo è rappresentato dal fatto che il disco System è protetto dalle copie in modo abbastanza efficace, tanto che i normali copiatori falliscono nell'intento. Inoltre Geos modifica la memoria del drive e se, durante l'utilizzo del programma, spegnete questa periferica per riaccenderla in seguito (allo scopo di tentare altre tecniche di duplicazione) non è poi possibile proseguire nell'utilizzo. Nemmeno disponendo della **cartuccia Mk6** sembrerebbe possibile effettuare copie di sicurezza. Con un accorgimento, però, è possibile sfruttare le potenzialità di questa cartuccia. Caricate Geos, come al solito, dal disco System; se volete, modificate la data ed il dispositivo di Input: i drivers, quindi, resteranno di default nella copia. Chiedete, ora, di formattare un disco e, dopo averne inserito uno, protetto in scrittura, battete un nome a caso in modo da costringere il drive ad emettere una segnalazione di errore. Ora potete, con Mk6, "**congelare**" la memoria e copiare anche resettando il drive. In questo modo, nel caricare il vostro file, il computer visualizzerà la finestra di errore che appariva al momento del "congelamento"; basterà ora inserire il disco di applicazioni desiderato e cliccare OK.

(Giuliano Simoncelli - Lizzana Rovereto)

Scuole con computers; dove sono?

Alcuni lettori, soprattutto studenti (ma c'è anche qualche insegnante) dichiarano di frequentare scuole pubbliche in cui sono disponibili (e, soprattutto, perfettamente funzionati) aule di informatica completamente attrezzate ma sfruttate poco (o per niente) a causa di una burocrazia macchinosa che tende ad evitare l'uso degli elaboratori per paura di romperli(!).
Dal momento che le "accuse" ci sembrano assurde (ma risultano, purtroppo, numerose) chiediamo la collaborazione dei nostri lettori per la stesura sistematica di una **"mappa" delle scuole italiane** in cui si verifica il fenomeno citato.

E' sufficiente scrivere indicandoci la scuola (indirizzo e numero di telefono, comprensivo di prefisso), il numero approssimativo degli studenti (e/o insegnanti) dell'intero complesso scolastico, il numero di aule di informatica espressamente attrezzate, il numero di computers installati per ciascuna aula ed il numero di ore settimanali durante i quali le aule vengono effettivamente adoperate.

Club di informatica; dove sono?

Agli albori degli anni '80, seguendo l'onda del boom informatico, sorsero decine di associazioni e clubs di informatica.
Alcuni di questi fallirono miseramente, altri furono addirittura costretti a limitare le iscrizioni che, in caso contrario, sarebbero risultate in numero eccessivo; molti furono i club che svolsero esclusivamente "servizi" di copiatura, pochi quelli in cui si sviluppò una cultura informatica nel senso ampio del termine.
Siamo quindi decisi a fare un **"censimento" di associazioni**, seriamente impegnate, da suggerire ai nostri lettori che dovessero abitare nelle vicinanze. Inviate, quindi, una lettera e, per dimostrare la validità del club, inserite anche uno o più bollettini ufficiali dell'associazione stessa o i volantini relativi alle attività svolte.

Inutile dire che, prima di pubblicare gli indirizzi, e le informazioni sulle caratteristiche dei clubs di informatica, ci accerteremo che questi non siano dei semplici... copiatori di videogames.

puter Club, i dischetti della serie Directory ed alcuni dei numerosissimi programmi professionali, vi troverete una vera e propria enciclopedia sul sistema 64.
Di più, onestamente, non si può fare...

Digitalizzatore vocale

Inserendo le due estremità di un microfono nel connettore joy (in corrispondenza delle prese per le paddle) dovrebbe esser possibile registrare variazioni di resistenza in funzione del suono captato dal microfono. Sviluppando l'idea (ed agendo sulla locazione 54297, corrispondente alla porta joy), è possibile realizzare un digitalizzatore vocale?
(Marco Carlotta - Caltanissetta)

Forse che sì, forse che no. In teoria tutto è possibile e, se non erro, tempo addietro fu commercializzato un prodotto h/w + s/w che era in grado, addirittura, di riconoscere alcuni comandi impartiti a viva voce.
Rivolgiamo, quindi, il quesito ai nostri lettori: c'è qualcuno in grado di portare a termine il progetto suggerito dal nostro Marco?

Concorso milionario

Scade il 15 marzo 1991 (quindi c'è tempo) il termine utile per la presentazione dei lavori di **Computer Animation** realizzati su Personal Computer all'edizione 1991 del premio **Riccione Bit Movie**.
Al primo premio verrà assegnata la cifra di un milione e mezzo di lire, al secondo un milione, al terzo mezzo milione.
La partecipazione presuppone una buona conoscenza della computer grafica e, comunque, la completa padronanza dei programmi di animazione diffusi tra gli appassionati.

Per informazioni telefonare a:
Centro della Pesa (041) 60.05.04 (ore 15:00 - 18:00)
Carlo Mainardi (541) 42.87.8
Stefano Leardini (0541) 37.73.88

(In)fedele nei secoli

Sono un 64-ista che, tra breve tempo, si procurerà un Amiga.
Scrivo per indurre gli altri lettori ad imitarmi, abbandonando il vecchio computer al quale non è necessario restare fedele perennemente.
So benissimo che il C/64 vanta migliaia di programmi, cartucce ed accessori vari (oltre al merito di aver fatto esplodere il Boom dell'informatica domestica:
senza il C/64 non ci sarebbe stata la diffusione della cultura informatica).
Però il tempo passa e la nostra rivista non può rischiare di chiamarsi "Cimitero Cimeli Commodore"...
(Andrea Tranchida - Enna)

Per rendersi conto della notevole differenza esistente tra i due computers, basta pensare che C/64 e drive costano 750 mila lire; un Amiga (e basta, perchè autosufficiente) solo 950 mila. E per 200 mila lire vale la pena rinunciare alle potenzialità di Amiga, anche solo per giocare? (figurarsi, poi, per lavorare...).
(Ikary - Arezzo)

Chissà perchè...

Ho un piccolo problema: possiedo un Commodore Vic 20, ma da molti anni non riesco più a trovare giochi su cassetta per questo computer.
Non risco a spiegarmi il perchè di questo fatto e, soprattutto, perchè la Commodore ha costruito e venduto un computer se sapeva che, un giorno, nessuno lo avrebbe più usato.
(Lettore fortunatamente anonimo)

Comprendo benissimo la tua perplessità, ma purtroppo il mondo è cattivo...

SYSTEMS EDITORIALE PER TE

La voce
Aggiunge al C/64 nuovi comandi Basic che consentono sia di far parlare il computer, sia di farlo Cantare! Diversi esempi allegati.

Cassetta: L. 12000 - Disco: L. 15000

Raffaello
Un programma completo per disegnare, a colori, con il C/64: linee, cerchi, quadrati, eccetera. Valido sia per disegno a mano libera che geometrico.

Cassetta: L. 10000

Oroscopo
Devi solo digitare la data di nascita e le coordinate geografiche del luogo che ti ha dato i natali. Vengono quindi elaborate le varie informazioni (case, influenze dei segni astrali, eccetera) e visualizzato un profilo del tuo carattere. Valido per qualsiasi anno, è indicato sia agli esperti sia ai meno introdotti. E' allegata una tabella delle coordinate delle più note città italiane e l'elenco delle ore legali in Italia dal 1916 al 1978.

Cassetta: L. 12000 - Disco: L. 12000

Computer Music
Cassetta contenente numerosi brani di successo da far eseguire, in interrupt, al tuo C/64 sfruttando, fino in fondo, il suo generatore sonoro (SID).

Cassetta: L. 12000

Gestione Familiare
Il più noto ed economico programma per controllare le spese e i guadagni di una famiglia.

Cassetta: L. 10000 - Disco: L. 10000

Banca Dati
Il più noto ed economico programma per gestire dati di qualsiasi natura.

Cassetta: L. 10000 - Disco: L. 10000

Matematica finanziaria
Un programma completo per la soluzione dei più frequenti problemi del settore.

Cassetta: L. 10000 - Disco: L. 20000

Analisi di bilancio
Uno strumento efficace per determinare con precisione i calcoli necessari ad un corretto bilancio.

Cassetta: L. 10000 - Disco: L. 20000

Corso di Basic
Confezione contenente quattro cassette per imparare velocemente le caratteristiche delle istruzioni Basic del C/64 e i rudimenti di programmazione. Interattivo.

Cassetta: L. 19000

Corso di Assembler
Un corso completo su cassetta per chi ha deciso di abbandonare il Basic del C/64 per addentrarsi nello studio delle potenzialità del microprocessore 6502. Interattivo.

Cassetta: L. 10000

Logo Systems
Il linguaggio più facile ed intuitivo esistente nel campo dell'informatica; ideale per far avvicinare i bambini al calcolatore.
Diversi esempi allegati.

Cassetta: L. 6500

Compilatore Grafico Matematico
Uno straordinario programma compilatore, di uso semplicissimo, che permette di tracciare, sul C/64, grafici matematici Hi-Res ad altissima velocità. Esempi d'uso allegati.

Cassetta: L. 8000

Emulatore Ms-Dos e Gw-Basic
Un prodotto, unico nel suo genere, che permette di usare, sul C/64 dotato di drive, la sintassi tipica del più diffuso sistema operativo del mondo. Ideale per studenti.

Solo su disco: L. 20000

Emulatore Turbo Pascal 64
Permette di usare le più importanti forme sintattiche del linguaggio Turbo Pascal (anche grafiche!) usando un semplice C/64 dotato di drive.
Ideale per studenti.

Disco: L. 19000

Speciale drive
Questo speciale fascicolo costituisce una guida di riferimento per le unità a disco del C64/128.
Comprende anche un velocissimo turbo-disk più la mappa completa della memoria del drive.

Fascicolo + disco: L. 12000

Utility 1
Un dischetto pieno zeppo di programmi speciali per chi opera frequentemente con il drive.

Disco: L. 12000

Utility 2
Seconda raccolta di utility indispensabili per realizzare sofisticate procedure di programmazione.

Disco: L. 15000

Graphic Expander 128
Per usare il C/128 (in modo 128 e su 80 colonne) in modo grafico Hi-res. Aggiunge nuove, potenti istruzioni Basic per disegnare in Hi-Res con la massima velocità in modalità 80 colonne.

Disco: L. 27000

Directory
Come è noto, a partire dal N. 10 di "Software Club" (la rivista su disco per l'utente dei "piccoli" computer Commodore), vengono riportati tutti i listati, in formato C/64-C/128, pubblicati su "Commodore Computer Club".
In precedenza tali listati venivano inseriti, mensilmente, in un dischetto, di nome "Directory", che oltre ai programmi di C.C.C. ospitava decine di altri file tra cui musiche nell'interrupt, giochi, listati inviati dai lettori e altro.
Ogni disco, dal prezzo irrisorio, contiene quindi una vera miniera di software. Ordinando i dischetti di "Directory" si tenga conto che al N. 1 corrispondeva il contenuto del N. 34 di "Commodore Computer Club", al N. 2 il N. 35 e così via.

Ogni dischetto: L. 10000

Super Tot '64
La nuova e completa edizione del programma Tot 13 con tutti i sistemi di riduzione e di condizionamento.
Ampia sezione dedicata alla teoria.

fascicolo + disco: L. 15000

Amiga Totospeed
Finalmente anche per Amiga un programma orientato alla compilazione delle schedine totocalcio.
Fai tredici conil tuo Amiga.

disco: L. 20000

SYSTEMS EDITORIALE PER TE

Disk'o'teca
Grazie a questa nutrita raccolta di brani musicali potrete divertirvi ascoltando i migliori brani prodotti dai vostri beniamini, oltre a una serie di composizioni prodotte "in casa".
In omaggio un bellissimo poster di Sting.
Disco: L.15.000

Assaggio di primavera
Esclusivo!
In un'unica confezione potrete trovare ben due cassette di videogiochi assieme a un comodo e funzionale joystick.
Cassette: L. 15.000

LIBRI TASCABILI

64 programmi per il C/64
Raccolta di programmi (giochi e utilità) semplici da digitare e da usare. Ideale per i principianti. **(126 pag.)**
L. 4800

I miei amici C/16 e Plus/4
Il volumetto, di facile apprendimento, rappresenta un vero e proprio mini-corso di Basic per i due computer Commodore. Numerosi programmi, di immediata digitazione, completano la parte teorica. **(127 pag.)**
L. 7000

62 programmi per C/16, Plus/4
Raccolta di numerosi programmi, molto brevi e semplici da digitare, per conoscere più a fondo il proprio elaboratore.
Ideale per i principianti. **(127 pag.)**
L. 6500

Micro Pascal 64
Descrizione accurata della sintassi usata dal linguaggio Pascal "classico". Completa il volume un programma di emulazione del PL/O sia in formato Microsoft sia in versione C/64 (da chiedere, a parte, su disco). **(125 pag.)**
L. 7000

Dal registratore al Drive
Esame accurato delle istruzioni relative alle due più popolari periferiche del C/64. Diversi programmi applicativi ed esempi d'uso. (94 pag.)
L. 7000

Il linguaggio Pascal
Esame approfondito della sintassi usata nel famoso compilatore. (112 pag.)
L. 5000

Simulazioni e test per la didattica
Raccolta di numerosi programmi che approfondiscono e tendono a completare la trattazione già affrontata sul precedente volume. **(127 pag.)**
L. 7000

Dizionario dell'Informatica
Dizionario inglese-italiano di tutti i termini usati nell'informatica. (Edizione completa). **(385 pag.)**
L. 10000

Word processing: istruzioni per l'uso
Raccolta delle principali istruzioni dei più diffusi programmi di w/p per i sistemi Ms-Dos: Word-Star, Samna, Multimate Advantage, Word 3. **(79 pag.)**
L. 5000

Unix
Un volumetto per saperne di più sul sistema operativo professionale per eccellenza.
Un necessario compendio per l'utente sia avanzato che inesperto **(91 pag.)**
L. 5000

ABBONAMENTO

Commodore Computer Club
11 fascicoli: L. 50.000

ARRETRATI

Ciascun numero arretrato di C.C.C. L. 6.000

Come richiedere i prodotti Systems

Coloro che desiderano procurarsi i prodotti della Systems Editoriale devono inviare, oltre alla cifra risultante dalla somma dei singoli prodotti, L. 3500 per spese di imballo e spedizione, oppure L. 6000 se si desidera la spedizione per mezzo raccomandata.
Le spese di imballo e spedizione sono a carico della Systems se ciascun ordine è pari ad almeno L. 50000.
Per gli ordini, compilare un normale modulo di C/C postale indirizzato a:

C/C Postale N. 37 95 22 07
Systems Editoriale Srl
Via Mosè, 22
20090 Opera (MI)

Non dimenticate di indicare chiaramente, sul retro del modulo (nello spazio indicato con "Causale del versamento"), non solo il vostro nominativo completo di recapito telefonico, ma anche i prodotti desiderati ed il tipo di spedizione da effettuare.
Per sveltire la procedura di spedizione sarebbe opportuno inviare, a parte, una lettera riassuntiva dell'ordine effettuato, allegando una fotocopia della ricevuta del versamento.
Chi volesse ricevere più celermente la confezione deve inviare la somma richiesta mediante assegno circolare, oppure normale assegno bancario (non trasferibile o barrato due volte) intestato a:

Systems Editoriale
Milano

di Alessandro de Simone

SCRIVO ANCH'IO? NO, TU NO

Alcuni listati inviati da vari lettori offrono lo spunto per esprimere le nostre opinioni sui programmi "self-made"; e per suggerire qualche idea su come migliorarli

Nonostante i numerosi inviti a telefonare in Redazione, molti sono gli aspiranti collaboratori che ignorano tale, preziosa, richiesta, ed inviano lavori di vario tipo.

Ma anche tra coloro che hanno preventivamente concordato la pubblicazione c'è chi non si attiene rigorosamente alle regole esposte, ed invia un lavoro sostanzialmente diverso da quanto concordato.

Per meglio far comprendere il nostro punto di vista cercheremo, in queste pagine, di far capire che cosa può interessare; e a chi.

La cassetta è morta

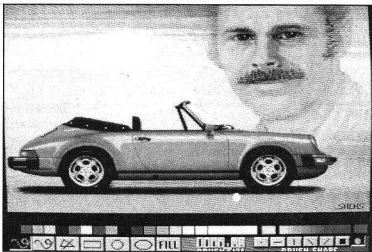
Il notevole lavoro di **Paolo Buratti** dimostra senza ombra di dubbio che, come egli stesso afferma, conosce molto bene il linguaggio macchina del **C/64**. Il programma inviato su cassetta, dal nome **Easy Debugger**, consente di rintracciare con facilità eventuali errori commessi durante la stesura di programmi l.m.

Grazie a un sistema di finestre, infatti, è possibile visualizzare un'area di memoria in formato esadecimale ed Ascii ed il corrispondente disassemblato; è presente, inoltre, una finestra di Help ed una di Input.

Il programma tiene conto di numerosi casi particolari (zona di memoria in Ram, Rom e sotto di esse) e consente l'output su stampante.

Purtroppo la notevole lunghezza dello stesso (**45** fogli di stampante dedicati al disassemblato!) impedisce la pubblicazione dello stesso sulla nostra rivista (chi mai se la sentirebbe di digitarlo?). Per non parlare del fatto che il lavoro è stato inviato su cassetta e, se non bastasse, non preventivamente concordato per telefono.

Qualcuno potrebbe obiettare che, almeno di fronte ai lavori più interessanti, noi della Redazione potremmo fare lo sforzo di inserire la cassetta nel registratore in modo da recuperare programmi e articoli. Il fatto è, cari lettori, che a causa del ridottissimo spazio a nostra disposizione siamo stati costretti ad eliminare tutto il materiale superfluo. Ne consegue che abbiamo a disposizione, relegato in un angolino, un solo **C/128-D**, privo di registratore e stampante, il cui schermo, per di più, viene condiviso da un sistema Ms-Dos che lo utilizza prepotentemente.

Non c'è più la possibilità fisica, quindi, di "trattare" nastri.

Tornando al programma inviato, pertanto, riteniamo che siano improponibili lavori di tal genere dal momento che, entro la fine del 1991, nessuno dei nostri lettori userà il C/64; e, se lo farà, non certo per digitare lunghissimi programmi, ma solo per sperimentare tecniche di programmazione originali oppure insolite, purchè **brevi**.

I nostri complimenti, comunque, al nostro Paolo Buratti che, se passerà definitivamente ad Amiga (oppure Ms-Dos) sarà certamente in grado di sfruttare adeguatamente la notevole preparazione, di livello professionale, dovuta allo studio approfondito del C/64.

❏ ❐

Un modem troppo teorico

Una lettera davvero singolare quella di **Filippo (di Rovigo)**. Questi, sofisticando il mini-listato apparso sul **N. 75** (che consentiva l'uso sommario di un **Modem**) è pervenuto ad un programma della lunghezza di oltre quattro pagine che consentirebbe, all'Amiga, di stabilire contatti tramite l'uso del telefono.

La lettera, però, conclude...

"Il programma lo invio su disco, e non via modem, perchè non posseggo questo apparecchio e, di conseguenza, non sono sicuro del completo funzionamento del file che vi mando..."

Ma, dico io, come è possibile che, in pieno 1990, ci sia ancora qualcuno che scriva programmi che non può verificare direttamente? Soprattutto per ciò che riguarda il modem, poi, intervengono tanti fattori, disturbi SIP a parte, che costringono i neo utenti a numerosi esperimenti da effettuare con programmi già ampiamente sperimentati.

E' come se qualcuno tentasse di scrivere un programma di hard copy non avendo una stampante sulla quale sperimentare l'efficacia del procedimento studiato. A parte questo piccolo(!) particolare, caro Filippo, il programma inviato contiene alcuni dettagli che, allo stesso tempo, sono positivi e negativi ed **una dissertazione sul loro conto sarà preziosissima anche per i lettori che ci seguono abitualmente.**

Un programma va giudicato a seconda della sua "destinazione". Prendiamo, ad esempio, il word processor (commercializzato o di pubblico dominio) che utilizzi abitualmente. E' molto probabile che questo sfrutti in modo intensivo le finestre, consenta la modifica dello stile e dei colori, presupponga l'uso di numerosi drivers per altrettante stampanti ed offra, in definitiva, tanti di quei preziosismi che obbliga l'utente ad esclamare "Che bello!" con la bocca spalancata.

Immagina, ora, che lo stesso programma, invece di ritrovartelo comodamente già pronto su dischetto, tu lo debba digitare. Riesci ad immaginare la fatica che devi compiere per trascrivere, da rivista, le centinaia di istruzioni (qualunque sia il linguaggio adoperato, Amigabasic, Assebly oppure C)?

Supponi, ancora, che tu abbia deciso di trascrivere il programma perchè ti interessa entrare in possesso di un buon word processor oppure, magari, perchè tu voglia studiare un particolare segmento del programma che, ad esempio, divida in sillabe una parola oppure consenta la giustificazione del testo.

In questi casi, ovviamente, non ti interessa l'opzione per modificare il colore del puntatore del mouse o la creazione di una finestra in cui far apparire l'eventuale messaggio di errore; ma presti attenzione, giustamente, al nocciolo del problema.

Purtroppo, di solito, il lettore "medio" non ha la necessaria competenza per individuare, tra i vari segmenti di programma pubblicati su una rivista, quello che più gli interessa; ne consegue che è obbligato a digitare l'intero listato, testarlo e, solo in seguito, studiare i vari segmenti che gli interessano.

Il lettore "medio", pertanto, tende a scoraggiarsi di fronte ai listati lunghi e complessi, anche se validissimi e perfettamente funzionanti; tra l'altro non ha nemmeno la garanzia che, una volta compiuta la fatica, il risultato sia all'altezza delle aspettative.

La conclusione?

Gli aspiranti collaboratori devono assolutamente evitare i comandi e le istruzioni di "abbellimento". Ad esempio, evitate il ricorso all'istruzione Palette (a meno che, ovviamente, non si tratti di un articolo dedicato al suo studio...); rinunciate alla creazione di finestre (se non strettamente indispensabili) che non fanno altro che rubar righe e creare difficoltà in fase di input dei dati; ricorrete alla creazione di menu di scelta con una sintassi che consenta di risparmiar righe e, contemporaneamente, insegni qualcosa su come raggiungere gli stessi risultati con un po' di fantasia. Esaminiamo, ad esempio, il programma "Crea menu" per Amiga, riportato nel riquadro e, dato che ci siamo, vediamo in che modo è possibile risparmiare righe da digitare pur sacrificando l'estetica.

Il primo **Input**, grazie ad una sola coppia di righe, consente di deviare il programma opportunamente.

La subroutine **vecchio** risulta, a onor del vero, decisamente chiara da interpretare; tuttavia il secondo metodo per creare un menu (**nuovo**) consente di sfruttare una banale procedura per l'allocazione di sei stringhe nelle rispettive posizioni di un menu a tendina. Si noti, addirittura, il ricorso alla fusione di stringhe (**a$ + " riga"**) per evitare di digitare più caratteri!

```
' Crea menu (AmigaBasic)
INPUT "(1) Vecchio (2) Nuovo metodo";a$
IF a$="1" THEN vecchio: ELSE GOTO nuovo

vecchio:
MENU 1, 0, 1, "prima riga"
MENU 1, 1, 1, "seconda riga"
MENU 1, 2, 1, "terza riga"
MENU 1, 3, 1, "quarta riga"
MENU 1, 4, 1, "quinta riga"
MENU 1, 5, 1, "sesta riga"
MENU ON: GOTO vedimenu

nuovo:
DATA 1-ma, 2-a, 3-za, 4-ta, 5-ta, 6-ta
FOR i=0 TO 5: READ a$: a$=a$+" riga"
MENU 1, i, 1, a$
NEXT : MENU ON: GOTO vedimenu

vedimenu:
PRINT "premi il puls. destro del mouse"
PRINT "un tasto per finire"
vai:
a$=INKEY$ :IF a$="" THEN vai
```

La visualizzazione di poco estetici messaggi ("premi un tasto...", eccetera) consente ulteriori risparmi di battitura di comandi (forse) esteticamente migliori (Print at, cambiamenti di colore, formattazioni del testo su video e così via) ma, soprattutto, la minore possibilità di commettere errori grazie alla visualizzazione non critica del messaggio stesso ed alla modifica che il lettore potrà pur sempre fare per proprio conto in fase di trascrizione del programma.

Si noti, infine, l'utilizzo di variabili dal nome semplice, se non semplicissimo (**A$, Vai, Nuovo, Vecchio**).

Il nome di subroutine **Vedimenu**, in quest'ottica, può risultare addirittura lunghissimo.

Non credano, i nostri lettori, che la tendenza al risparmio di caratteri da digitare sia, alla fin fine, controproducente per la chiarezza del programma.

Prendiamo il caso della variabile stringa da assegnare con Input (vedi lo stesso programma citato) che può essere definita, indifferentemente, con **A$** oppure con...

TastoPremutoInFaseDiInput$

Nel secondo caso, ovviamente, il nome della variabile ricorda con estrema chiarezza la funzione che svolge. Se, però, ci mettiamo nei panni del lettore che trascrive il listato, è facile preventivare un errore di battitura.

Ne consegue che la variabile stringa, durante l'elaborazione del programma che la utilizza, potrebbe contenere una stringa diversa da ciò che serve, con le conseguenze che è facile immaginare; e tutto ciò, badate bene, senza che il computer emetta una segnalazione di errore. Un brutale esempio chiarirà meglio il concetto.

IF TastoP**remuto**InFaseDiInput$ = "a" THEN...

IF TastoP**remto**InFaseDiInput$ = "a" THEN

E' facile intuire che, a causa della mancanza della "**u**" (in ...**EMTO**..., nel secondo caso) il programma seguirà percorsi diversi da quanto preventivato e, comunque, difficilissimi da rintracciare.

Non è molto meglio un banale...

IF A$ = "a" THEN

...che limita errori di trascrizione e di controllo?

In conclusione:

Mettetevi nei panni dei lettori e tenete conto che questi digitano volentieri i listati, soprattutto se brevi e contenenti solo tutto ciò che serve per raggiungere lo scopo prefissato nell'articolo.

Una volta trascritto il programma, sarà lo stesso lettore a decidere l'introduzione di abbellimenti, finestre, suoni, variabili con altro nome e così via. Allo stesso modo si esamini il programma **Ongosub** che evita la catena di istruzioni del tipo **If... Then... Gosub** facendo ricorso alla più comoda ed efficace forma sintattica **On... Gosub**.

I programmi inviati, se rispettassero i semplici principi esposti prima, sarebbero enormemente più brevi e avrebbero una maggior probabilità di essere accettati per la pubblicazione.

Colori in libertà

Rosario De Chiara invia un programma per **Amiga** in grado di creare dei "pulsanti" sul video e di gestire il flusso del lisitato a seconda di quale di essi venga cliccato con il puntatore del mouse. Il programma utilizza tecniche di programmazione che tengono conto dei suggerimenti segnalati nel precedente paragrafo; tuttavia la routine non è di impiego universale (come la desiderano i nostri lettori) e rimane limitata ad un caso piuttosto particolare: la visualizzazione di soli sei pulsanti, di dimensione ben precisa, nella parte inferiore del video. Come fare per crearne di più? Il programma "Supermenu programmabile" (vedi CCC N. **77**), ad esempio, affronta e risolve il problema della gestione di un qualsiasi numero di pulsanti (compatibilmente con la dimensione del video, ovviamente) posizionati ovunque nello schermo e contenenti, o meno, dei messaggi.

Al lettore "medio", insoma, interessa studiare una tecnica di programmazione quanto più **versatile** possibile, in modo da **adattarla** in propri listati. A che può servire una routine specifica per visualizzare sei (e solo sei) bottoni?

Se ciò non bastasse, inoltre, il programma inviato risulta piuttosto lento e la routine che modifica il colore con l'istruzione Palette sembra non funzionare sempre e a volte genera l'errore "Illegal

```
inizio:
INPUT "(1) RoutineA. (2) RoutineB. (3) RoutineC"; a$
' On gosub...
ON VAL(a$) GOSUB RoutineA, RoutineB, RoutineC: GOTO inizio

' If Then...
IF a$="1" THEN GOSUB RoutineA
IF a$="2" THEN GOSUB RoutineB
IF a$="3" THEN GOSUB RoutineC
GOTO inizio

RoutineA:
PRINT Questa e' la routine A"
RETURN

RoutineB:
PRINT "Questa e' la routine B"
RETURN

RoutineC:
PRINT "Questa e' la routine C"
RETURN
```

Un suggerimento per evitare la "catena" di If ... Then

Function Call". Che ci vuole ad inserire un controllo di errore?

Rosario, tuttavia, dimostra di essersi impegnato a fondo e di conoscere abbastanza bene l'uso corretto di variabili e subroutines.

À che serve?

Anche nel caso di **Mauro Barilli** ci troviamo di fronte ad un **esperto** Amigo che, però, ha forse dimenticato i tempi duri in cui, anche per lui, il computer sembrava parlare turco, o quasi.

Ci propone, infatti, un programma in **AmigaBasic** in grado di modificare il numero dei bitplanes disponibili. La scoperta dell'utilità di un simile listato viene però affidata agli stessi lettori dal momento che, come dimostrativo, il programma si limita a visualizzare un numero di triangoli colorati proporzionale al numero dei bitplanes attivati.

Se non bastasse, l'articolo avverte che *"...possono verificarsi Guru Meditation con alcuni programmi che condividono l'area di Workbench"*. Mancano, però, esempi di riferimento che, di certo, renderebbero più interessante il lavoro svolto dal lettore.

A chi è dunque destinato il programma? Non certo agli esperti, che queste cose le sanno già; ma nemmeno ai principianti, a causa della terminologia difficile, della mancanza di spiegazioni di alcuni termini apparentemente elementari (**librerie, bitmaps, allocazione in memoria** e così via) e, soprattutto, di applicazioni pratiche. Basterebbe un ciclo **For ... Next** incaricato di creare rettangoli pieni ed una spiegazione del perchè non è possibile visualizzare più di quattro colori con due soli bitplanes.

Un riferimento sul conseguente rallentamento dell'elaborazione non guasterebbe e metterebbe in guardia i lettori da un'apparente... staticità!

L'articolo, poi, è stato scritto non tenendo assolutamente conto che un **word processor** è l'esatto contrario di **una macchina da scrivere**.

Alla fine del rigo video, infatti, non bisógna inserire un trattino per la divisione in sillabe o, peggio, un ritorno carrello. Allo stesso modo bisogna pensare che i files che inviate vengono "trattati" da altri sistemi di impaginazione che non usano l'Amiga, bensì un sistema Ms-Dos. Per non cadere vittime della Babele di codici, è assolutamente indispensabile evitare accuratamente le vocali accentate, che non hanno (a quanto pare) un codice internazionale. Bisogna, ad esempio, scrivere **a'** e non **à**. Evitare anche i Tab ed i simboli particolari, soprattutto quelli il cui codice Ascii è superiore al valore **127**.

Il programma inviato, tuttavia, è una piccola miniera di idee e spunti da cui trarre articoli di notevole interesse, se presentati uno alla volta.

Fatti sentire per telefono, stavolta, e vedrai che concorderemo insieme un argomento da sviluppare per conto di CCC.

Grafica povera

Anche il programma inviato da **Maurizio Ramondo** è scritto in **Amigabasic** e sembra destinato ai principianti. In effetti l'articolo risulta ben scritto (finalmente qualcuno che sa usare un word processor!) ed il listato è pieno di Rem esplicative, abbastanza chiare.

E' possibile, date le coordinate dei vertici di una figura, far apparire la figura stessa in tre dimensioni, ruotarla secondo uno dei tre assi, ingrandirla, ridurla e così via.

Purtroppo il listato non contiene quegli accorgimenti che, in un programma di grafica, sono un'esigenza ormai irrinunciabile.

Ad esempio, manca un controllo sulla effettiva visualizzazione della figura; può capitare, infatti, che questa sia al di fuori dell'area visibile. Ma questo è il meno.

La rotazione è possibile solo a "scatti" di 90 gradi lungo i tre assi: ne consegue che un cubo, ad esempio, rimane sempre un cubo e solo figure complesse danno una prova tangibile dell'avvenuta rotazione. In effetti il lettore precisa che la scelta della rotazione a scatti di 90 gradi è dovuta alla volontà di non complicare il programma con la trigonometria. Mah!...

E allora nasce ancora spontanea la domanda: a che serve un programma di grafica quando l'articolo, pur se correttamente impostato, non spiega in che modo applicare le formule matematiche che consentono la visualizzazione in tre dimensioni partendo dalle coordinate dei singoli vertici?

L'uso delle subroutine, e la suddivisione razionale dei vari compiti da svolgere, dimostrano tuttavia che Maurizio Ramondo ha una certa confidenza con l'AmigaBasic e che, ne sono sicuro, non passerà molto tempo prima che ci invii qualche lavoro decisamente utile per i nostri lettori.

Andrea Megalini propone un listato, sempre in **AmigaBasic**, che effettivamente sembra non avere le carenze prima accennate. Il programma consente di effettuare rotazioni di **qualsiasi angolo** lungo uno dei tre assi x, y oppure z, permette di creare, registrare, caricare e modificare figure geometriche da realizzare con l'unione punto - per - punto. L'articolo, però, è decisamente **avaro di particolari** e mancano le **Rem** compensative nel programma. Questo non ricorre a nessun preziosismo tecnico e, di conseguenza, **c'è un mare di istruzioni che si potrebbero evitare**. Dulcis in fundo... il word processor per scrivere l'articolo è stato usato al peggio delle sue possibiltà.

La stoffa del programmatore, nonostante tutto, non manca al nostro Andrea Megalini che preferisce, evidentemente, "tradurre" formule matematiche in AmigaBasic senza curarsi troppo dell'efficienza della programmazione. Fatti sentire, Andrea: con un po' di pazienza, potrai vedere qualche tuo lavoro sulle nostre pagine...

Ancòra grafica

Giosuè **Cozzolino** invia un programma, in **AmigaBasic**, che visualizza 24 "bottoni", (4 righe x 8 colonne) colorati in modo random. Selezionandone uno con il mouse è possibile variare le percentuali di **rosso verde** e **blu** mediante altrettanti slides da far scorrere ancòra con l'ausilio del mouse (più o meno come per il menu **Preferences**, quando si selezionano i colori dello schermo).

L'articolo di accompagnamento (a proposito: quando ci decideremo ad usare un word processor in modo adeguato?) si limita ad accennare il modo di

```
Rosso:
WHILE MOUSE (0) = - 1
mx = MOUSE (1)

IF mx < x1 AND x1 > 10 THEN
colore (0, cl) = colore (0, cl) - cs
PALETTE cl, colore (0, cl), colore (1, cl), colore (2, cl)
LINE (x1, y2 + 1) - STEP (4, 8), 0, bf
x1 = INT (10 + 160 * colore (0, cl))
LINE (x1, y2 + 1) - STEP (4, 8), 1, bf
LINE (112, y2 + 50) - STEP (20, 20), cl, bf
ELSEIF mx > x1 + 8 AND x1 < 170 THEN
colore (0, cl) = colore (0, cl) + cs
PALETTE cl, colore (0, cl), colore (1, cl), colore (2, cl)
LINE (x1, y2 + 1) - STEP (4, 8), 0, bf
x1 = INT (10 + 160 * colore (0, cl))
LINE (x1, y2 + 1) - STEP (4, 8), 1, bf
LINE (112, y2 + 50) - STEP (20, 20), cl, bf
END IF

WEND

RETURN
```

La subroutine colore. Nel listato inviato da Giosuè Cozzolino vi sono altre due routines praticamente identiche, che rappresentano un inutile doppione

```
x1 = icsA: x2 = icsB: nc = 0: gosub colorato
x1 = icsC: x2 = icsC: nc = 1: gosub colorato
x1 = icsD: x2 = icsE: nc = 2: gosub colorato

colorato:
WHILE MOUSE (0) = - 1
mx = MOUSE (1)
IF mx < x1 AND x1 > 10 THEN
colore (nc, cl) = colore (nc, cl) - cs
PALETTE cl, colore (0, cl), colore (1, cl), colore (2, cl)
LINE (x1, y2 + 1) - STEP (4, 8), 0, bf
x1 = INT (10 + 160 * colore (nc, cl))
LINE (x1, y2 + 1) - STEP (4, 8), 1, bf
LINE (112, y2 + 50) - STEP (20, 20), cl, bf
ELSEIF mx > x1 + 8 AND x1 < 170 THEN
colore (0, cl) = colore (0, cl) + cs
PALETTE cl, colore (0, cl), colore (1, cl), colore (2, cl)
LINE (x1, y2 + 1) - STEP (4, 8), 0, bf
x1 = INT (10 + 160 * colore (0, cl))
LINE (x1, y2 + 1) - STEP (4, 8), 1, bf
LINE (112, y2 + 50) - STEP (20, 20), cl, bf
END IF
WEND: RETURN
```

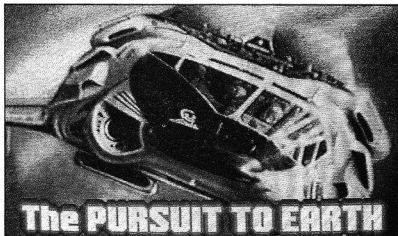
La nostra proposta di modifica. Come si può intuire (anche se non è pubblicato l'intero listato), sono sufficienti alcune assegnazioni di variabili prima di richiamare un'unica routine, di nome **Colorato**.

utilizzare il programma. A chi, però, può interessare far apparire 24 bottoni e modificarne il colore?

Se non bastasse, il programma, pur se ben strutturato, spesso ricorre a ripetizioni di subroutines che potrebbero essere evitate ricorrendo alla modifica di alcune variabili.

Ad esempio, la subroutine di nome **Rosso** è sostanzialmente identica alle altre due (di nome **Verde** e **Blu**) tranne che per piccoli particolari.

Invece di scrivere...
GOSUB rosso
GOSUB verde
GOSUB blu
...si potrebbero impostare un paio di parametri prima di effettuare il salto, come indicato nel riquadro a parte. Il vantaggio?

Un programma più breve (anche e soprattutto, da digitare...). Coraggio, ancora un po' di esperienza e diventerai un ottimo programmatore.

AGUZZA L'INGEGNO!

Era parecchio tempo che non sfidavamo i nostri affezionati lettori; stavolta, addirittura, lanciamo due guanti...

Chi usa abitualmente i computer professionali (Amiga ed Ms- Dos, ovviamente) spesso possiede una miriade di programmi ed utility di notevole interesse, ma non riesce a sfruttarli al massimo delle loro potenzialità.

Uno dei principali desideri degli utenti cosiddetti "evoluti" consiste nel fondere insieme le potenzialità di più programmi o di inserirli in **file batch** che li richiamino in modo rapido ed efficace.

Purtroppo la... mancanza di fantasia non sempre consente di sviluppare procedure di un certo interesse, sia perchè queste si riferiscono a configurazioni molto particolari, sia perchè difficilmente si riscono ad immaginare facilitazioni non previste dal programma o utility, prese in considerazione.

Ne consegue che l'unico file batch scritto dall'utente è solo quello relativo all'inizializzazione del sistema; quello, cioè, che viene attivato automaticamente non appena si accende il computer. Altri file batch, in effetti, vengono portati a termine, ma quasi sempre funzionano soltanto sul computer dell'utente a causa della presenza di utility dislocate in directory ben precise.

Quanto tempo è passato?

Tutti sanno che l'esperienza di un pilota di aerei viene brevemente riassunta dal numero di ore di volo al suo attivo: un pilota che vanta 5000 ore di volo è sicuramante più "affidabile" di un novellino con sole 30 ore.

La stessa cosa potrebbe valere per un più modesto automobilista il quale, però, può solo vantare un certo numero di anni di possesso della patente. Un pilota di aerei, infatti, può essere preciso nel dichiarare le ore di volo effettuate perchè la burocrazia aeronautica è, grazie a Dio, molto pignola al riguardo. Tutte le volte che un aereo decolla, un addetto alla torre di controllo annota con cura l'orario di partenza e, nell'aeroporto di arrivo, c'è un altro controllore di volo che annota l'orario di atterraggio. Viaggiando in automobile, invece, nessuno annota a che ora si mette in moto il motore e quella in cui parcheggiamo. Un domani, chissà, inventeranno una patente magnetica che, inserita al posto della chiave nel cruscotto, memorizzerà tutti i dati relativi al percorso effettuato.Tornando al presente, invece, ecco la nostra idea:

Scrivere una procedura Batch (o di altro tipo) che consenta di creare un file Ascii che, continuamente aggiornato, consenta di determinare il tempo dedicato all'uso di un particolare programma professionale.

In questo modo, insomma, sarà possibile vantare un certo numero di ore di lavoro su un word processor, un data base, un DTP, uno spreadsheet.

"*Pensa* - direte ai vostri amici - *al mio attivo ho 340 ore di Wordstar; posso dire, senza vantarmi(!) di conoscerlo abbastanza bene* ".

In pratica si tratta di scrivere un file batch che, come primo gruppo di istruzioni, aggiunga, in coda al file incaricato di memorizzare tutti i tempi precedentemente trascorsi usando quel particolare programma, la data e l'ora "attuali".

Come secondo gruppo di istruzioni conterrà l'attivazione del programma di cui si vuol registare il tempo di attività e, come terzo ed ultimo gruppo, la memorizzazione, sullo stesso file di prima, della data e dell'ora di "chiusura" della procedura.

Il tutto, ovviamente, dovrà essere gestito automaticamente dal computer e

l'intera procedura dovrà risultare "trasparente" all'utente stesso.

Naturalmente sarà necessario scrivere, a parte, un programma (in un qualsiasi linguaggio) che sia in grado di interpretare correttamente il file che contiene tutti i dati e di eseguire la somma dei tempi trascorsi, espressa in ore.

Piccola, grande grafica

I moderni elaboratori, e soprattutto l'**Amiga**, offrono potenzialità grafiche inimmaginabili fino a qualche tempo fa. E' infatti possibile ammirare, su video e a colori, stupende riproduzioni di foto, di quadri o di immagini digitalizzate; per non parlare dei disegni sviluppati autonomamente dagli utenti-artisti.

Purtroppo la quantità di informazioni necessarie per memorizzare una schermata, specie se a colori, è decisamente rilevante; ne consegue che, spesso, su un dischetto si possono memorizzare pochissime immagini.

Come partecipare

Gli argomenti proposti in queste pagine, come si intuisce facilmente, sono sviluppabili solo da chi usa abitualmente calcolatori **Amiga** o **Ms-Dos** compatibili.

Il **C/64** ed il **C/128**, infatti, non consentono di approfondire con sufficiente semplicità una procedura in grado di portare a termine i compiti proposti.

Non ci stancheremo di ripetere che, **prima** di inviare il frutto delle vostre fatiche, è assolutamente **indispensabile** contattare telefonicamente la Redazione per verificare se, in effetti, il lavoro svolto merita la divulgazione sulle nostre pagine.

Si tenga presente che, in ogni caso, verranno privilegiati programmi e procedure che si distinguano per la loro semplicità e, soprattutto, **brevità** di digitazione.

Programmi troppo lunghi verranno scartati per ovvi motivi.

L'eventuale compenso viene stabilito volta per volta secondo insindacabili criteri stabiliti dalla Redazione.

Fortunatamente sono a disposizione efficaci e rapidissimi programmi compattatori che, grazie a sofisticate tecniche informatiche, riducono la dimensione di un qualsiasi file (vedi, ad esempio, **"I compressori di Amiga"**, C.C.C. n. **74**).

La dimensione finale di un file, però, varia a seconda dei casi ed, in particolare, è tanto minore quanto maggiore è la ripetitività di byte posti in successione. Spieghiamoci meglio.

Consideriamo il seguente programma in Gw-Basic...

```
100 OPEN "file-a" FOR OUTPUT AS #1
120 FOR I=1 TO 1000
130 PRINT #1, "1";: PRINT "1";
140 NEXT: CLOSE #1
```

...che non fa altro che scrivere 1000 caratteri **"1"** sul file di nome **"File-a"**.

Il seguente programma, invece...

```
100 OPEN "file-ab" FOR OUTPUT AS
#1
120 FOR I=1 TO 1000
125 A$=RIGHT$ (STR$ (INT (RND*10)
),1)
130 PRINT #1, A$;: PRINT A$;
140 NEXT: CLOSE #1
```

...scrive ancora 1000 caratteri numerici, solo che, stavolta, sono casuali.

La lunghezza dei due files, però, è sempre la stessa:

1001 bytes su disco.

Se ora, con il famoso compattatore **Pkzip**, li compattiamo entrambi, notiamo con sorpresa che il primo file (quello di soli "1", per intendeci) occuperà solo **149** bytes; il secondo, invece, ben **831** (questo valore potrà esser diverso sul vostro computer).

Il motivo della notevole diversità risiede nella stessa procedura di compattazione con cui opera Pkzip.

Di solito, infatti, un compattatore riduce il numero di bytes se questi sono quasi tutti eguali tra loro e posti l'uno dopo l'altro.

Maggiore è la diversità (ed il "disordine"), minore sarà la compattazione possibile (e viceversa).

Il caso limite, ovviamente, è rappresentato da un file interamente costituito da byte tutti eguali tra loro, come il file **"Prova-a"**.

In che cosa, dunque, può consistere una sfida legata a questo argomento?

Semplice: se effettuate varie prove di compattazione di files grafici, realizzati con il vostro programma grafico preferito, vi accorgerete che, ad esempio, le schermate costituite da un certo numero di linee orizzontali presentano un numero di bytes diverso che se fossero costituite da un ugual numero di linee verticali, di eguale lunghezza.

Analogamente un elevato numero di colori influisce sulla lunghezza finale del file, compattato o meno che sia.

La sfida, quindi, si rivolge a coloro che si sentono un po' artisti, un po' ricercatori.

Si tratta, anzitutto, di stabilire con quali "trucchi" è possibile risparmiar bytes; e già questo, da solo, può esser lo spunto per la stesura di un articolo.

Impadronitisi della tecnica di "riduzione", quindi, si tratta di inviarci schermate grafiche di "effetto" che richiedano la minima occupazione su disco.

I lettori potranno (o meglio, dovranno) modificare le diffusissime immagini digitalizzate (che chiunque di noi possiede) in modo da sfruttare le potenzialità di riduzione scoperte: eliminazione di uno o più colori, modifica o eliminazione di alcune linee o aree che creino difficoltà in fase di compattazione e così via.

L'immagine finale, ovviamente, dovrà essere sufficientemente rassomigliante all'originale: non inviate una riproduzione de "La Gioconda" in cui si vede solo il sorriso su fondo rosso...

Naturalmente potranno sbizzarrirsi anche gli artisti in erba che siano in grado di creare immagini di un certo effetto nonostante ricorrano alle limitazioni imposte da una efficace compattazione.

Non dimenticate che, in ogni caso, lo scopo finale deve essere quello di **limitare al massimo il numero dei bytes** di ciascuna schermata in modo che, dopo opportuna de-compattazione, sia possibile, ad esempio, visualizzare in successione numerose schermate senza essere costretti a cambiar dischetto dopo poche immagini.

Anche in questo caso, prima di inviare il vostro lavoro (da sviluppare esclusivamente su computer Amiga oppure Ms-Dos compatibili) è indispensabile contattare telefonicamente la Redazione per una prima verifica.

CAMPUS

SOMMARIO

18 - CARO FILE, TI SCRIVO

Un listato nel doppio formato Basic (per chi desidera digitare ed impartire il Run, senza crearsi problemi) e Macro Assembler Commodore (per chi intende approfondire lo studio del linguaggio macchina). Questo mese proponiamo la sostituzione del comando Stop con Type, presente nei sistemi più evoluti come l'Amiga e l'Ms-Dos. Il comando implementato, inoltre, consente di "spezzare" file Ascii che risultassero troppo lunghi per essere caricati dal vostro Word Processor preferito. Numerose sono le routine l.m. utilizzate, come pure i "salti" al Kernal ed i trucchi di programmazione.

29 - ADATTATORE 6499 + TELEFONO, FELICE UNIONE

La bolletta del telefono cresce in maniera inversamente proporzionale alla velocità di trasmissione dei dati. Chi si è reso conto di questa triste realtà forse ha rinunciato, da tempo, ad utilizzare l'adattatore telematico 6499 che rappresentò, anni fa, il primo gradino nel mondo della rice/trasmissione dei dati. Se non volete rinunciare al C/64, tirate fuori dalla naftalina il vetusto accessorio: potrete, magari, utilizzarlo per scrivere un programma di gestione delle conversazioni telefoniche, in tempo reale, senza effettuare alcun collegamento elettronico; un esauriente articolo sulla manipolazione dei due relais presenti nel 6499.

di Filippo Bruno

CARO FILE, TI SCRIVO...

Un comando molto utile in ambiente Amiga ed Ms-Dos, ma rigorosamente assente sul C/64, è Type, che permette di esaminare il contenuto di un file presente su disco

Un'occasione d'oro per chi vuole cimentarsi con il linguaggio macchina del C/64

Più di una volta, lavorando sul C/64, ho avvertito la necessità di utilizzare comandi disponibili su sistemi più evoluti, quali **Ms-Dos** ed **Amiga**, rimanendo non solo deluso, ma anche notevolmente disorientato riguardo la via da intraprendere per superare vari inconvenienti.

Un comando molto utile in ambiente Ms-Dos, e rigorosamente assente sul C/64, è Type, che permette di esaminare il contenuto di un file presente su disco trasportando su video ogni carattere che lo costituisce.

Type è molto utile, ad esempio, per esaminare un file sequenziale generato con un word processor, salvato con un nome strano o troppo breve(!), senza essere costretti a caricare il programma per la gestione dei testi.

Presi quindi la decisione di supplire alla carenza del Basic 2.0 e di implementare il comando type sul C/64. Tanto più che si sarebbero potuti leggere anche i files generati da **Macroassembler**, tutti rigorosamente in formato **SEQuenziale**. Ma, come è noto, un abbellimento tira l'altro e così mi sono ritrovato a realizzare un comando capace anche di duplicare un file sotto altro nome, o addirittura **spezzarlo** in due parti. Così, aspettando che iniziasse la scuola, dopo ben due giorni di orario continuato sulla tastiera, smanettando tra cartucce, monitor, assembler, loader, dischi e soprattutto calmanti, sono riuscito a portare a termine il lavoro.

In queste pagine troverete un po' di tutto: riferimenti al **Kernal**, all'interprete **Basic**, accesso al disco in linguaggio macchina, inserimento di un comando Basic personalizzato, gestione della **pagina zero** (quella da $0000 a $0100); forse manca qualche manipolazione del raster register, ma non sono proprio riuscito ad inserirla...

A parte gli scherzi, cominciamo subito la trattazione.

Ai vostri comandi

Per implementare un nuovo comando Basic è innnanzitutto necessario avere bene in mente il compito che verrà chiamato a svolgere.

Si dovrà quindi elaborare un'apposita routine, inevitabilmente in lm, ed allocarla da qualche parte in memoria, più precisamente in una zona

libera da possibili influenze da parte di Basic, variabili, gestione del registratore (ma buttatelo via, una volta per tutte!), memoria video, puntatori in pagina 0 e chi ne ha più ne metta.

Perchè per forza in lm? Se riflettete, comprenderete che il calcolatore "ragiona" a base di **0** e di **1** (linguaggio binario), ossia a seconda della tensione assente o presente all'interno dei sui circuiti. Tutti saprete(?) che il Basic è un linguaggio interprete, il che sta a significare che anche quando noi scriviamo il classico **Print "pippo"**, il computer svolge il compito mandando in esecuzione un programma in lm di decine di bytes, allocato nella Rom.

La routine lm

La routine di cui ci occupiamo questo mese è allocata a partire da **$C000** (dec. 49152), la classica zona Ram libera da interferenze, e la sua sintassi è:

Sys 49152 "pippo" [,"poppo" [,"peppo"]]

Le parentesi quadre stanno a significare che il loro contenuto si può omettere, dando luogo, ovviamente, a risultati diversi. Vi sono tre modi di utilizzarla. Digitando...

Sys 49152 "pippo"

...il drive comincerà a ronzare e visualizzerà sullo schermo il contenuto del file "pippo" (se, ovviamente, esiste su disco).

Premendo il tasto **Return**, la stampa si interromperà fino a che non verrà premuto nuovamente.

Se invece, dopo il primo return, verrà premuto il tasto **F7**, il file verrà chiuso e si tornerà al Basic. Digitando...

Sys 49152 "pippo", "poppo"

...il file di nome "pippo" verrà visualizzato sullo schermo e, contemporaneamente, verrà duplicato su disco sotto il nome di "poppo". Valgono le stesse le considerazioni sui tasti da premere.

La terza sintassi...

Sys 49152 "pippo", "poppo", "peppo"

...si otterrà lo stesso effetto della seconda. Se, però, dopo aver premuto return premerete **F1**, il file "poppo" verrà **chiuso** e ne verrà **aperto** un altro col nome di "peppo", che continuerà a ricevere dati dal file "pippo". In pratica otterremo ancora una copia del file originale ("pippo"), ma formata da due files, contenenti, rispettivamente, la parte memorizzata fino alla pressione di F1, e quella invece memorizzata in seguito alla pressione stessa.

La terza sintassi sarà molto utile per "spezzare" files troppo lunghi, magari ricevuti via interfaccia da un sistema più grande, in modo da essere trattati anche con un w/p del C/64.

Easy Script, ad esempio, può elaborare poco(!) più di 700 righe (o 24000 caratteri) ed elimina la parte terminale di documenti che risultassero più lunghi.

Il disassemblato

Il disassemblato è sufficientemente commentato per essere compreso anche da chi è entrato da poco nell'affascinante mondo del linguaggio macchina. Tuttavia si rende necessaria qualche spiegazione in più.

Vediamo innanzitutto la procedura da adottare allorchè si intenda aprire un file su disco. In Basic, per aprire un file, occorre dichiarare il **numero** del canale, la **periferica**, l'indirizzo **secondario**, nonchè il **nome** del file, il **tipo**, abbinato al **modo** di colloquio (se si intende leggerne uno già esistente o se si vuole crearne uno ex- novo). In lm occorre inserire questi dati (ma non tutti) dalla locazione $B7 a $BC.

In **$B7** va memorizzata la lunghezza del nome del file, in **$B8** il numero del canale, in **$B9** l'indirizzo secondario (è compreso tra 2 e 14, mentre 5 è usato per inviare un comando al drive e 0 e 1 sono usati dalla Cpu per le operazioni di Load & Save); in **$BA** la periferica su cui aprire il file, mentre in **$BB** e **$BC** vanno immessi, rispettivamente, il byte basso e il byte alto della locazione di memoria ove reperire il nome.

Per aprire il file, poi, basta una chiamata all'indirizzo **$F34A**, tramite Jsr, ed quivalente al GoSub del Basic. Per chiudere un file occorre caricare l'**acc**umulatore con il numero del canale da chiudere, e poi saltare a **$FFC3**, sempre tramite Jsr.

Occorre, ora, stabilire se aprire un file in **input** oppure in **output**. Si carica, a tale scopo, il registro **X** con il numero del canale, e si salta, rispettivamente, a **$FFC6** (mnemonico CHKIN) oppure a **$FFC9** (mnem. CHKOUT).

Per prelevare un carattere, o meglio un byte, occorre saltare ad un altro indirizzo, **$FFCF** (mnem. CHRIN), mentre per inviarlo, si usa un salto a **$FFD2** (mnem. CHROUT). Occorre fare

La relativa brevità del listato non tragga in inganno il lettore: il nuovo comando "Type" è di notevole potenza

Nelle routines pubblicate c'è di tutto: attivazione Ram posta sotto la Rom, sostituzione di un comando Basic...

moltissima attenzione all'uso delle ultime due routine, perchè in assenza dei parametri di Open, prelevano o scaricano un carattere sulla periferica **0**, quella di default, ossia la tastiera ed il video.

Potrebbe capitarvi, in caso di errore, di osservare sullo schermo un bel cursore che ripete ciò che avete digitato prima di premere return.

Per facilitare il compito viene utilizzata un'altra routine del Kernal, allocata in **$FFCC** (mnem. CLRCHN) che si occupa di ripristinare i valori standard di input / output alla tastiera ed al video, pulendone i canali.

Nel "far pulizia" dopo ogni lettura e scrittura, si è ovviamente costretti a dichiarare più volte il tipo dei file (input / output), ma in questo modo la Cpu non si confonde con i canali e non rischia di aprire, in lettura, quello sbagliato.

Un ultimo cenno sulla routine in **$FFE7** (mnem. CLALL) che si occupa di chiudere tutti i files eventualmente aperti. Sempre in tema di drive, da notare l'utilizzo della variabile riservata Basic **ST**, che altri non è se non la locazione **144** ($90), che restituisce **0** se è tutto ok, **64** se il file è finito e non vi sono più caratteri da leggere; se non ci fosse un controllo di questo tipo, il drive continuerebbe a leggere (e magari duplicare) una infinita serie di shift + g, simbolo della fine del file.

Nel Kernal esiste anche un indirizzo (**$FFB7**, mnem. READST) che si occupa di leggere lo stato ST del file e memorizzarlo in accumulatore, ma è più immediato (e veloce) un semplice LDA $90 piuttosto che un salto alla subroutine.

Dopo la prima Open indirizzata al canale di comando, e la seconda al file da leggere, la routine controlla se esso esiste o meno, o comunque se si sono verificati errori, magari causati da manomissione di tracce, protezioni che generano errori, disco non ancora formattato, apertura dello sportello e simili amenità.

Questo avviene leggendo i primi due caratteri dello status del drive dal canale di comando. Se sono entrambi degli zeri allora tutto va bene, altrimenti stampa l'intero messaggio. Se si fosse verificato, ad esempio, un "File not found", il cui codice di errore è **62**, la Cpu visualizzerebbe l'intero messaggio e scriverebbe ",file not found,"..., in quanto abbiamo già letto i primi due numeri.

Così occorre memorizzarli da qualche parte e stamparli richiamandoli da Ram.

Ma se non si fosse verificato un errore, alla fine della procedura il drive riscriverebbe l'intera sequenza di numeri, in quanto rappresenterebbe la conferma della riuscita di altre operazioni, e non del controllo della esistenza del file "pippo". Perciò otterremmo "**0000**, ok, 00, 00", ossia 4 zeri invece dei soliti due.

Di conseguenza, se il file esiste, vengono resettati i due byte in pagina zero dove erano memorizzate le prime due cifre del codice di errore, in modo che quando la Cpu stamperà due **Chr$(0)**, non visualizzerà, di fatto, nulla.

Passiamo ora ad analizzare la procedura attraverso la quale si legge una stringa.

Saltando a **$AD9E**, la Cpu controlla la presenza di una stringa posta dopo l'ultimo comando impartito da Basic (la ben nota Sys 49152) e aggiorna il contatore di caratteri per far sì che, ad un successivo controllo (ad esempio la presenza di un'altra stringa), non venga controllata sempre la stessa, ma quella che, eventualmente, si trova dopo di essa.

Non trovandola, emette una segnalazione di errore. Posto che ci sia una stringa (nel nostro caso "pippo"), saltando a **$B6A3** la Cpu preleva i dati che la riguardano; più precisamente, verrà posta nell'accumulatore la lunghezza della stringa, mentre in $22 / $23 il byte basso / alto della zona di memoria dove reperirne il contenuto.

Facendo una prova preliminare, si nota che in condizioni di normalità la Cpu alloca i caratteri che formano la stringa in fondo alla memoria Basic, dove di solito deposita le stringhe dichiarate nei programmi, solo che non aggiorna i puntatori in pagina zero ($33 / $34).

Il Top di memoria

La scelta di questa zona comporta un altro problema, perchè aggiungendo al secondo nome ("poppo") il suffisso "**, s, w**" per generare un file sequenziale in scrittura, non possiamo mettere in coda i quattro caratteri (le due virgole, la "s" e la "w"), in quanto invaderemmo l'area compresa tra **$A000** e **$BFFF**, che la Cpu gestisce in modo singolare (vedi la seconda parte di questo articolo).

Siamo quindi costretti a traslare di quattro posti i dati della stinga verso sinistra, ossia verso l'inizio del Basic, aggiornarne i puntatori ($22 / $23) ed aumentare la lunghezza di quattro valori.

Infatti le stringhe, a mano a mano che vengono dichiarate, vengono automaticamente memorizzate a partire da **$9FFF** verso **$0801** (inizio Basic) fino a che la fine del programma Basic coincida con l'inizio delle stringhe, nel qual caso l'immissione di un'altra stringa provoca il famigerato *?out of memory error*.

Ecco giustificata, quindi, la sfilza di INC $22 e DEC $22 (occupano solo un byte in più rispetto ad un ciclo chiamato da tre subroutine).

```
100 REM IMPLEMENTAZIONE DI "TYPE"
110 REM BY FILIPPO BRUNO (C) 1990
120 REM SOLO PER C/64
125 CT=0:PRINT"▮LEGGO DATI LM..."
127 POKE 1,55
130 READ A:IF A= -1 THEN 150
140 POKE49152+CT,A:CT=CT+1
145 CK=CK+A:GOTO130
150 IF CK<>58725THEN PRINT"ERRORE":END
160 FORI=0TO23:READA:X=X+A:POKE832+I,A
162 NEXT:IFX<>4347THENPRINT"ERR.2":END
165 PRINT"TRASFERISCO LA ROM IN RAM"
170 SYS 832
180 POKE 41004,255: POKE 41005,191
190 POKE 41183,84: POKE 41184,89
195 POKE 41185,80: POKE 41186,197
200 PRINT"FATTO!": POKE 1,54
210 END
1000 DATA 032,122,192,032,140,192,032
1010 DATA 161,192,032,178,192,032,071
1020 DATA 193,165,250,201,048,208,041
1030 DATA 165,251,201,048,208,035,032
1040 DATA 196,192,162,008,032,224,192
1050 DATA 165,248,032,210,255,165,247
1060 DATA 240,005,162,009,032,240,192
1070 DATA 032,022,193,165,249,240,230
1080 DATA 169,000,133,250,133,251,032
1090 DATA 204,255,169,008,032,195,255
1100 DATA 169,009,032,195,255,169,141
1110 DATA 032,210,255,032,210,255,165
1120 DATA 250,032,210,255,165,251,032
1130 DATA 210,255,162,015,032,198,255
1140 DATA 032,207,255,032,210,255,165
1150 DATA 144,240,246,032,204,255,032
1160 DATA 140,192,032,231,255,169,000
1170 DATA 133,198,096,169,015,133,184
1180 DATA 133,185,169,008,133,186,169
1190 DATA 000,133,183,032,074,243,096
1200 DATA 032,204,255,162,015,032,201
1210 DATA 255,169,073,032,210,255,032
1220 DATA 204,255,169,000,133,247,096
1230 DATA 032,158,173,032,163,182,166
1240 DATA 034,164,035,201,016,144,002
1250 DATA 169,016,096,133,183,169,008
1260 DATA 133,184,133,185,133,186,134
1270 DATA 187,132,188,032,074,243,096
1280 DATA 032,121,000,201,044,208,020
1290 DATA 032,115,000,230,247,032,252
1300 DATA 192,032,121,000,201,044,208
1310 DATA 005,032,115,000,230,247,096
1320 DATA 032,198,255,032,207,255,133
1330 DATA 248,165,144,133,249,032,204
1340 DATA 255,096,032,201,255,165,248
1350 DATA 032,210,255,032,204,255,096
1360 DATA 032,161,192,032,090,193,133
1370 DATA 183,169,009,133,184,133,185
1380 DATA 169,008,133,186,134,187,132
1390 DATA 188,032,074,243,096,165,197
1400 DATA 201,001,208,042,032,156,193
1410 DATA 165,197,201,001,240,030,201
1420 DATA 003,208,006,032,056,192,076
1430 DATA 116,164,201,004,208,236,165
1440 DATA 247,201,002,208,230,198,247
1450 DATA 169,009,032,195,255,032,252
1460 DATA 192,032,156,193,096,162,015
1470 DATA 032,198,255,032,207,255,133
1480 DATA 250,032,207,255,133,251,032
1490 DATA 204,255,096,133,252,160,000
1500 DATA 177,034,198,034,198,034,198
1510 DATA 034,198,034,145,034,230,034
1520 DATA 230,034,230,034,230,034,200
1530 DATA 196,252,208,231,198,034,198
1540 DATA 034,198,034,198,034,169,044
1550 DATA 145,034,200,169,083,145,034
1560 DATA 200,169,044,145,034,200,169
1570 DATA 087,145,034,024,165,252,105
1580 DATA 004,166,034,164,035,096,160
1590 DATA 016,162,000,202,208,253,136
1600 DATA 208,248,096,-1
2000 DATA 162,032,169,000,168,133,252
2010 DATA 169,160,133,253,177,252,145
2020 DATA 252,200,208,249,230,253,202
2030 DATA 208,244,096
2040 END

READY.
```

Non vengono effettuati controlli per verificare se occorre decrementare anche il byte alto ($23, che di norma contiene il valore $9F) allorchè il valore in $22 diventi "negativo" (il termine è tra virgolette perchè in linguaggio macchina, quando si decrementa di una unità una locazione che contiene 0, assume il valore 255).

Si presuppongono, quindi, condizioni di default trascurando di considerare il caso (raro) in cui si voglia elaborare una stringa lunga più di 255 caratteri.

Questo controllo si dovrebbe però effettuare (riscrivendo parte della routine) nel caso in cui si abbia l'esigenza di spostare altrove l'indirizzo di fine Basic ($37 / $38) per esigenze personali (quali la salvaguardia di una zona destinata ai caratteri ridefiniti, sprite, etc...).

Il controllo sul **tasto premuto** si ottiene esaminando la locazione **197** ($C5) che conterrà **1** se il tasto è return, **3** se il tasto è F7, **4** se il tasto è F1.

La pausa si rende necessaria (anche se non risolve sempre tutti i problemi) perchè la velocità di esecuzione, molto maggiore di quella dell'utente che preme il tasto return, porterebbe la Cpu a pensare che il suddetto tasto sia stato premuto più volte; e se il numero di volte risultasse pari... sarebbe come se non fosse stato premuto affatto.

Come controllare la presenza di altre stringhe? Non possiamo controllare la virgola con la nota subroutine in $AEFD, in quanto genererebbe un *?syntax error* se non la trovasse (e potrebbe essere così, dato che gli altri due nomi sono facoltativi).

Sono state utilizzate due routine lm (in verità la seconda è la stessa, ma lanciata un tratto dopo) del Basic, allocate in pagina zero. Con la

...controlli di I/O, segnalazioni di errore, duplicazione di files ed altro!

seconda, situata in **$0079**, si preleva il carattere dopo l'ultima istruzione (sarebbe Sys 49152, ma avendo letto la prima stinga, il contatore di caratteri punta a ciò che viene dopo la prima stringa) senza aggiornare il contatore caratteri. Ciò fa sì che se il carattere non è una virgola (o non c'è affatto) non venga generato alcun errore. In caso contrario (se la virgola è presente), si provvede ad incrementare il suddetto contatore rileggendo il carattere, ma con la routine allocata in **$0073** (mnem. CHRGET) in modo da leggere ciò che viene dopo. Un'ultima cosa da notare è la presenza di due shift + return (ascii = **141**) posti dopo la chiusura dei file, per evitare che il messaggio sullo status del drive compaia affiancato all'ultimo byte letto e visualizzato.

Inoltre viene pulito il buffer di tastiera per evitare la visualizzazione di eventuali return premuti durante la visualizzazione.

Comandi

Passiamo ora alla procedura per implementare un nuovo comando Basic. Dal momento che l'argomento non viene affrontato da molto tempo, vedremo come sostituire, ad un comando Basic poco usato (quali **Let** oppure **Stop**), uno a nostro piacimento (ma c'è un limite a tutto...). L'interprete Basic, che tra l'altro è allocato proprio da **$A000** a **$BFFF**, contiene tutto il lm necessario per "capire" il Basic ed agire di conseguenza.

Perciò i progettisti del C/64 hanno inserito, negli 8k di Rom, anche le **tabelle** utili sia per il riconoscimento dei comandi sia per sapere dove reperire le singole routine di gestione.

La prima tabella si trova da **41118** a **41373** (256 dati) e contiene le parole riservate del Basic, tranne **pi**-greco, **St**, **Ti$** e **Ti,** che sono elaborate diversamente. Ogni parola è separata daAlla successiva proprio dall'ultimo carattere, il cui **Msb** (**m**ost **s**ignificant **b**it, bit più significativo), cioè il settimo, è posto ad uno, con la conseguenza che l'ultimo carattere della parola- chiave ha il codice Ascii "normale" sommato a 128.

Nella tabella pubblicata **non** sono stati riportati i valori da 41257 in poi, in quanto appartengono a parole chiave quali funzioni matematiche o stringa, che non si prestano ad essere modificate, dal momento che presuppongono, tra l'altro, altre parole chiave o caratteri speciali (tra cui l'uguale) prima di esse. Rispettivamente, da 40972 a 41040 (il taglio è riferito alla prima tabella), è presente un'altra tabella contenente gli indirizzi di salto, nel solito, consueto, abituale, formato byte basso / alto.

Questa volta, però, tanto per non fare le cose troppo uguali, l'indirizzo che punta alle routine

end	41118	40972
for	41121	40974
next	41124	40976
data	41128	40978
input#	41132	40980
input	41138	40982
dim	41143	40984
read	41146	40986
let	41150	40988
goto	41153	40990
run	41157	40992
if	41160	40994
restore	41162	40996
gosub	41169	40998
return	41174	41000
rem	41180	41002
stop	41183	41004
on	41187	41006
wait	41189	41008
load	41193	41010
save	41197	41012
verify	41201	41014
def	41207	41016
poke	41210	41018
print#	41214	41020
print	41220	41022
cont	41225	41024
list	41229	41026
clr	41233	41028
cmd	41236	41030
sys	41239	41032
open	41242	41034
close	41246	41036
get	41251	41038
new	41254	41040

Tabella degli indirizzi
Il primo valore indica il primo byte in cui è memorizzato il nome dell'istruzione; il secondo, invece, l'inizio della routine di interpretazione del comando stesso.

va diminuito di uno. Per far puntare l'istruzione **Let** a **$403B** (ricordatevi di abbassare il top di memoria per evitare sovrapposizioni con il Basic) bisognerà trascrivere, in 40988 / 40989, $3a e $40 rispettivamente. A prima vista la cosa sembra facile: basta cambiare la tabella dei nomi, quella dei salti ed il gioco è fatto. Ma come si fa a scrivere su Rom?

Da Rom a Ram

Guarda caso, però, sappiamo che "sotto" gli 8k di Rom ve ne sono altri 8 di Ram, inutilizzati (se non da videogiochi di un certo livello), e che se leggiamo un dato da quella

zona, leggiamo da Rom, mentre se lo scriviamo, scriviamo su Ram (e, quanto a razionalità, ci sembra più che giusto...).

Sappiamo, inoltre, che la **locazione 1** in **pagina zero** ha il compito, fra l'altro, di comunicare alla Cpu (tramite il suo **primo bit**) se prendere in considerazione la Rom oppure la Ram presente da $A000 a $BFFF (si veda l'articolo "Se la memoria non vacilla", apparso su **C.C.C. n. 53** pagina 71). Lasciando inalterato, nella locazione 1, il valore **55 di default**, la Cpu elabora i dati considerando attiva la Rom dell'interprete Basic. Resettando il primo bit, cioè portando il valore del byte a **54**, la Cpu attiverà la Ram sottostante. Provate a digitare...

Poke 1, 54

...ed osservate che succede, magari dopo aver trascritto il listato senza salvarlo! Il computer si blocca, costringendovi a spegnerlo e riaccenderlo, a meno che non possediate un pulsante esterno di reset.

Questo perchè la Cpu ritiene di avere sott'occhio l'interprete Basic (la parola magica Poke viene interpretata proprio grazie alla presenza del Basic); solo che nella Ram sottostante c'è un'accozzaglia casuale di schifezze, ed il computer entra in collasso.

Un semplice ciclo for... next, del tipo...

for i = 40960 to 49152: poke i, peek (i): next

...seguito da una...

Poke 1, 54

...risolverebbe il problema.

Ma se non volete addormentarvi mentre il computer trasferisce gli **8192** dati, vi consigliamo di copiare il listato in lm, che fa le stesse cose del ciclo For... Next del Basic, solo che impiega un tempo leggermente inferiore... A questo punto ricordatevi di scegliere un nome della stessa lunghezza di quello che volete cambiare e di aggiungere **128** al codice Ascii dell'ultimo carattere. Cambiate anche i puntatori di salto, inserendo nel nostro caso $FF e $BF, dato che 49152 - 1 = 49151 = $bfff.

Dovendo scegliere tra Let e Stop, abbiamo preferito Stop perchè è di 4 caratteri: un comando **typ** suonava male. Se non volete rinunciare a Stop potete sempre cambiare Let in **Alt**, per esempio, e sostituire i puntatori di Let con quelli di Stop, letti prima di effettuare Poke 1, 54.

Il programma caricatore pensa ad allocare la routine a partire da $C000, a cambiare Rom in Ram, nome e puntatori di salto, per avere un comando dalla sintassi più professionale:

type "pippo" [,"poppo" [,"peppo"]]

Finalmente abbiamo raggiunto il nostro scopo

Concludendo

La routine si presta ad eventuali modifiche, quali l'uscita del file letto non su video ma su **stampante**, magari specificando, al momento di impartire il comando, la periferica voluta. Se però provate a stampare un file PRG su carta, il codice #$08, sempre presente all'inizio di un file di tipo Prg (rappresenta, infatti, il primo byte dell'indirizzo di inizio Basic $0801) provocherà l'entrata in modo grafico della stampante, con conseguenze più che imprevedibili.

Per come è stata progettata, la routine lanciata nel modo 2 (oppure 3) duplica un file sempre in modo **SEQ**uenziale, perchè un programma Basic tiene conto anche di link di linea, numero linee, byte iniziali per sapere dove allocarlo, etc., il che renderebbe impossibile lo sdoppiamento, senza errori, di un file PRG in due PRG.

Digitate con attenzione il listato che comprende anche la routine di "copia" del Basic su Ram

```
ROM->RAM......PAGE 0001

LINE# LOC   CODE       LINE

00001 0000                   *=$0340
00002 0340  A2 20            LDX #$20       ;INPOSTA $FC/$FD
00003 0342  A9 00            LDA #$00       ;COME PUNTATORI ALL'AREA
00004 0344  A8               TAY            ;A PARTIRE DA $A000
00005 0345  85 FC            STA $FC        ;E TRASFERISCE I DATI
00006 0347  A9 A0            LDA #$A0       ;DA ROM A RAM,
00007 0349  85 FD            STA $FD        ;TRASFERENDO PER 32 VOLTE
00008 034B  B1 FC      LOOP  LDA ($FC),Y    ;(X=$20) 256 DATI
00009 034D  91 FC            STA ($FC),Y    ;(0<Y<255)
00010 034F  C8               INY            ;INCREMENTANDO DI VOLTA
00011 0350  D0 F9            BNE LOOP       ;IN VOLTA IL BYTE ALTO
00012 0352  E6 FD            INC $FD        ;DI TALE ZONA
00013 0354  CA               DEX            ;
00014 0355  D0 F4            BNE LOOP       ;
00015 0357  60               RTS            ;
00016 0358                   .END
```

```
TYPE/64......PAGE 0001

LINE# LOC   CODE        LINE


00001  0000                        OPEN     = $F34A
00002  0000                        CHKIN    = $FFC6
00003  0000                        CHKOUT   = $FFC9
00004  0000                        CHRIN    = $FFCF
00005  0000                        CHROUT   = $FFD2
00006  0000                        CLOSE    = $FFC3
00007  0000                        CLALL    = $FFE7
00008  0000                        CLRCHN   = $FFCC
00009  0000                 ;------------------------------
00010  0000                        GETSTR   = $AD9E
00011  0000                        STRPUN   = $B6A3
00012  0000                        BASIC    = $A474
00013  0000                        CHRGET   = $0073
00014  0000                        CHRGE2   = $0079
00015  0000                        ST        = $90
00016  0000                        KEYBRD    = $C5
00017  0000                        FLEN      = $B7
00018  0000                        FNUM     = $B8
00019  0000                        SADDR    = $B9
00020  0000                        DEVICE   = $BA
00021  0000                        FNAME    = $BB
00022  0000                        MODE     = $F7
00023  0000                        DATO     = $F8
00024  0000                        COPYST   = $F9
00025  0000                        NUMERR   = $FA
00026  0000                        LEN      = $FC
00027  0000                 ;------------------------------
00028  0000                        *= $C000
00029  C000  20 7A C0              JSR OPEN15      ;APRE IL CANALE DI COMANDO
00030  C003  20 8C C0              JSR INIT        ;INIZIALIZZA IL DRIVE
00031  C006  20 A1 C0              JSR STRING      ;PRELEVA UNA STRINGA
00032  C009  20 B2 C0              JSR OPEN8       ;APRE UN FILE SUL CANALE 8
00033  C00C  20 47 C1              JSR FOUND       ;CONTROLLA SE ESISTE:
00034  C00F  A5 FA                 LDA NUMERR      ;LEGGE I PRIMI DUE CARATTERI
00035  C011  C9 30                 CMP #$30        ;DELLO STATUS DEL DRIVE
00036  C013  D0 29                 BNE NOTFND      ;SE SONO DIVERSI DA 48 (0)
00037  C015  A5 FB                 LDA NUMERR+1;ALLORA STAMPA L'ERRORE
00038  C017  C9 30                 CMP #$30        ;(PRESUMIBILMENTE FILE
00039  C019  D0 23                 BNE NOTFND      ;NOT FOUND)
00040  C01B  20 C4 C0              JSR ALTRAS      ;C'E' UN'ALTRA STRINGA?
00041  C01E  A2 08          TRASF  LDX #$08        ;CANALE 8 IN LETTURA
00042  C020  20 E0 C0              JSR LEGGE       ;LEGGE UN DATO E LO DEPOSITA
00043  C023  A5 F8                 LDA DATO        ;IN PAGINA 0,DA CUI LO RICARICA
00044  C025  20 D2 FF              JSR CHROUT      ;E LO INVIA SU VIDEO
00045  C028  A5 F7                 LDA MODE        ;MODO 1?
00046  C02A  F0 05                 BEQ MODE1       ;SI->SALTA
00047  C02C  A2 09                 LDX #$09        ;NO->CANALE 9 IN SCRITTURA
00048  C02E  20 F0 C0              JSR SCRIVE      ;LO SCRIVE SU DISCO
00049  C031  20 16 C1       MODE1  JSR RETURN      ;RETURN PREMUTO?
00050  C034  A5 F9                 LDA COPYST      ;LEGGE LA COPIA DELLO STATUS
00051  C036  F0 E6                 BEQ TRASF       ;SE TUTTO BENE RICOMINCIA
00052  C038  A9 00          FINE   LDA #$00        ;IL FILE E' FINITO:RESETTA
00053  C03A  85 FA                 STA NUMERR      ;I 2 BYTE COL NUMERO DELL'ERRORE
00054  C03C  85 FB                 STA NUMERR+1    ;(PER NON VISUALIZZARE 0000,OK..
00055  C03E  20 CC FF       NOTFND JSR CLRCHN      ;AL POSTO DI 00,OK..)-RESET CANA
LI


TYPE/64......PAGE 0002
```

```
LINE# LOC   CODE          LINE

00056  C041  A9 08            LDA #$08        ;CHIUDE IL CANALE 8
00057  C043  20 C3 FF         JSR CLOSE       ;
00058  C046  A9 09            LDA #$09        ;CHIUDE IL CANALE 9
00059  C048  20 C3 FF         JSR CLOSE       ;
00060  C04B  A9 8D            LDA #141        ;STAMPA DUE SHIFT+RETURN
00061  C04D  20 D2 FF         JSR CHROUT      ;PER NON FAR VENIRE ATTACCATO
00062  C050  20 D2 FF         JSR CHROUT      ;LO STATUS CON L'ULTIMO BYTE TES
TO
00063  C053  A5 FA            LDA NUMERR      ;STAMPA IL NUMERO DELL'ERRORE
00064  C055  20 D2 FF         JSR CHROUT      ;(SE 0 NON SI VEDE PERCHE'
00065  C058  A5 FB            LDA NUMERR+1    ;CHR$(0) NON VIENE VISUALIZZATO)
00066  C05A  20 D2 FF         JSR CHROUT      ;
00067  C05D  A2 0F            LDX #$0F        ;CANALE DI COMANDO (15)
00068  C05F  20 C6 FF         JSR CHKIN       ;IN LETTURA
00069  C062  20 CF FF  ANCORA JSR CHRIN       ;PRENDE UN CARATTERE
00070  C065  20 D2 FF         JSR CHROUT      ;LO STAMPA SU VIDEO
00071  C068  A5 90            LDA ST          ;CONTROLLA SE I CARATTERI
00072  C06A  F0 F6            BEQ ANCORA      ;DELLO STATUS SONO FINITI
00073  C06C  20 CC FF         JSR CLRCHN      ;RESETTA I CANALI I/O
00074  C06F  20 8C C0         JSR INIT        ;INIZIALIZZA IL DRIVE
00075  C072  20 E7 FF         JSR CLALL       ;CHIUDE TUTTI I FILE/CANALI APER
TI
00076  C075  A9 00            LDA #$00        ;PULISCE IL BUFFER
00077  C077  85 C6            STA $C6         ;DI TASTIERA
00078  C079  60               RTS
00079  C07A                   ;--------------------------------
00080  C07A  A9 0F     OPEN15 LDA #$0F        ;PREDISPONE CANALE 15
00081  C07C  85 B8            STA FNUM        ;CON INDIRIZZO
00082  C07E  85 B9            STA SADDR       ;SECONDARIO 15
00083  C080  A9 08            LDA #$08        ;PERIFERICA = 8
00084  C082  85 BA            STA DEVICE      ;
00085  C084  A9 00            LDA #$00        ;LUNGHEZZA NOME FILE=0
00086  C086  85 B7            STA FLEN        ;(NON C'E')
00087  C088  20 4A F3         JSR OPEN        ;LO APRE
00088  C08B  60               RTS             ;
00089  C08C                   ;--------------------------------
00090  C08C  20 CC FF  INIT   JSR CLRCHN      ;RESETTA I CANALI I/O
00091  C08F  A2 0F            LDX #$0F        ;CANALE 15
00092  C091  20 C9 FF         JSR CHKOUT      ;IN SCRITTURA
00093  C094  A9 49            LDA #$49        ;INVIA IL CARATTERE
00094  C096  20 D2 FF         JSR CHROUT      ;'I' (ASCII = $49)
00095  C099  20 CC FF         JSR CLRCHN      ;RESETTA I CANALI I/O
00096  C09C  A9 00            LDA #$00        ;AZZERA IL FLAG DEL MODO
00097  C09E  85 F7            STA MODE        ;
00098  C0A0  60               RTS             ;
00099  C0A1                   ;--------------------------------
00100  C0A1  20 9E AD  STRING JSR GETSTR      ;CONTROLLA LA STINGA
00101  C0A4  20 A3 B6         JSR STRPUN      ;NE PRELEVA LUNGHEZZA E PUNTATOR
I
00102  C0A7  A6 22            LDX $22         ;PUNTATORI IN $22/$23
00103  C0A9  A4 23            LDY $23         ;(FORMATO BYTE BASSO/ALTO)
00104  C0AB  C9 10            CMP #$10        ;E LUNGHEZZA IN ACCUMULATORE
00105  C0AD  90 02            BCC OK          ;SE ACC. >16 -> TRONCA
00106  C0AF  A9 10            LDA #$10        ;LA STRINGA
00107  C0B1  60        OK     RTS             ;
00108  C0B2                   ;--------------------------------
00109  C0B2  85 B7     OPEN8  STA FLEN        ;ACCUMULATORE IN FLEN
00110  C0B4  A9 08            LDA #$08        ;PREDISPONE IL NUMERO 8 COME
```

```
TYPE/64......PAGE 0003

LINE# LOC   CODE        LINE

00111  C0B6  85 B8             STA FNUM      ;CANALE DI UN FILE,
00112  C0B8  85 B9             STA SADDR     ;INDIRIZZO SECONDARIO,
00113  C0BA  85 BA             STA DEVICE    ;E PERIFERICA
00114  C0BC  86 BB             STX FNAME     ;SCRIVE PUNTATORI NOME FILE
00115  C0BE  84 BC             STY FNAME+1   ;CHE SONO ANCORA IN X E Y
00116  C0C0  20 4A F3          JSR OPEN      ;LO APRE
00117  C0C3  60                RTS           ;
00118  C0C4                ;-----------------------------------
00119  C0C4  20 79 00   ALTRAS JSR CHRGE2    ;CONTROLLA CARATTERE
00120  C0C7  C9 2C             CMP #$2C      ;E' UNA VIRGOLA? (ASCII = $2C)
00121  C0C9  D0 14             BNE TORNA     ;NO:TORNA
00122  C0CB  20 73 00          JSR CHRGET    ;SI:'SALTA' LA VIRGOLA
00123  C0CE  E6 F7             INC MODE      ;MODO = 2 (MODE = 1)
00124  C0D0  20 FC C0          JSR OPEN9     ;APRE IL CANALE 9
00125  C0D3  20 79 00          JSR CHRGE2    ;CONTROLLA CARATTERE
00126  C0D6  C9 2C             CMP #$2C      ;ANCORA UNA VIRGOLA?
00127  C0D8  D0 05             BNE TORNA     ;NO:TORNA
00128  C0DA  20 73 00          JSR CHRGET    ;SI:LA 'SALTA'
00129  C0DD  E6 F7             INC MODE      ;MODO = 3 (MODE = 2)
00130  C0DF  60         TORNA  RTS           ;
00131  C0E0                ;-----------------------------------
00132  C0E0  20 C6 FF   LEGGE  JSR CHKIN     ;FILE CANALE X IN LETTURA
00133  C0E3  20 CF FF          JSR CHRIN     ;PRENDE UN DATO
00134  C0E6  85 F8             STA DATO      ;LO MEMORIZZA PER ORA IN RAM
00135  C0E8  A5 90             LDA ST        ;MEMORIZZA LO STATUS
00136  C0EA  85 F9             STA COPYST    ;IN RAM
00137  C0EC  20 CC FF          JSR CLRCHN    ;RESETTA CANALI I/O
00138  C0EF  60                RTS           ;
00139  C0F0                ;-----------------------------------
00140  C0F0  20 C9 FF   SCRIVE JSR CHKOUT    ;FILE CANALE X IN SCRITTURA
00141  C0F3  A5 F8             LDA DATO      ;LEGGE DATO DA RAM
00142  C0F5  20 D2 FF          JSR CHROUT    ;STAMPA SU VIDEO
00143  C0F8  20 CC FF          JSR CLRCHN    ;RESETTA CANALI I/O
00144  C0FB  60                RTS           ;
00145  C0FC                ;-----------------------------------
00146  C0FC  20 A1 C0   OPEN9  JSR STRING    ;PRENDE UNA STRINGA
00147  C0FF  20 5A C1          JSR ADD       ;AGGIUNGE SUFFISSO ",S,W"
00148  C102  85 B7             STA FLEN      ;ACC. IN LUNGHEZZA STRINGA
00149  C104  A9 09             LDA #$09      ;PREDISPONE IL NUMERO 9 COME
00150  C106  85 B8             STA FNUM      ;CANALE E
00151  C108  85 B9             STA SADDR     ;INDIRIZZO SECONDARIO
00152  C10A  A9 08             LDA #$08      ;PERIFERICA = 8
00153  C10C  85 BA             STA DEVICE    ;
00154  C10E  86 BB             STX FNAME     ;SCRIVE I PUNTATORI DEL NOME
00155  C110  84 BC             STY FNAME+1   ;
00156  C112  20 4A F3          JSR OPEN      ;APRE IL FILE
00157  C115  60                RTS           ;
00158  C116                ;-----------------------------------
00159  C116  A5 C5      RETURN LDA KEYBRD    ;LEGGE DA TASTIERA:
00160  C118  C9 01             CMP #1        ;PREMUTO IL TASTO RETURN?
00161  C11A  D0 2A             BNE TORNA2    ;NO:TORNA SENZA PAUSA
00162  C11C  20 9C C1          JSR PAUSA     ;SI:PAUSA
00163  C11F  A5 C5      LP3    LDA KEYBRD    ;RILEGGE LA TASTIERA:
00164  C121  C9 01             CMP #1        ;RIPREMUTO IL TASTO RETURN?
00165  C123  F0 1E             BEQ TORNA1    ;SI: TORNA CON PAUSA
```

```
TYPE/64......PAGE 0004

LINE# LOC   CODE        LINE

00166  C125  C9 03            CMP #3         ;PREMUTO F7?
00167  C127  D0 06            BNE NOFINE     ;NO:NON FINIRE
00168  C129  20 38 C0         JSR FINE       ;SI:FINISCI
00169  C12C  4C 74 A4         JMP BASIC      ;E TORNA AL BASIC
00170  C12F  C9 04     NOFINE CMP #4         ;PREMUTO F1?
00171  C131  D0 EC            BNE LP3        ;NO:RICOMINCIA DA CAPO
00172  C133  A5 F7            LDA MODE       ;SI:MODO 3? (MODE = 2)
00173  C135  C9 02            CMP #$02       ;
00174  C137  D0 E6            BNE LP3        ;NO:RICOMINCIA
00175  C139  C6 F7            DEC MODE       ;SI:MODO = 2 (MODE = 1)
00176  C13B  A9 09            LDA #$09       ;CHIUDI IL CANALE 9
00177  C13D  20 C3 FF         JSR CLOSE      ;
00178  C140  20 FC C0         JSR OPEN9   E RIAPRILO CON L'ALTRO NOME
00179  C143  20 9C C1  TORNA1 JSR PAUSA      ;PAUSA
00180  C146  60        TORNA2 RTS
00181  C147            ;--------------------------------
00182  C147  A2 0F     FOUND  LDX #$0F       ;CANALE 15
00183  C149  20 C6 FF         JSR CHKIN      ;IN LETTURA
00184  C14C  20 CF FF         JSR CHRIN      ;PRELEVA 2 CARATTERI
00185  C14F  85 FA            STA NUMERR     ;E LI MEMORIZZA
00186  C151  20 CF FF         JSR CHRIN      ;IN RAM IN DUE BYTE
00187  C154  85 FB            STA NUMERR+1   ;IN PAGINA 0
00188  C156  20 CC FF         JSR CLRCHN     ;RESETTA I CANALI I/O
00189  C159  60               RTS            ;
00190  C15A            ;--------------------------------
00191  C15A  85 FC     ADD    STA LEN        ;ACC.(LUNGHEZZA) IN RAM
00192  C15C  A0 00            LDY #$00       ;CONTATORE = 0
00193  C15E  B1 22     TRASLA LDA ($22),Y    ;CARICA IL PRIMO CARATTERE DEL N
OME
00194  C160  C6 22            DEC $22        ;E,SOTTRAENDO 4 NUMERI
00195  C162  C6 22            DEC $22        ;AL PUNATORE DEL NOME,
00196  C164  C6 22            DEC $22        ;LO RISCRIVE 4 BYTE
00197  C166  C6 22            DEC $22        ;PIU' INDIETRO
00198  C168  91 22            STA ($22),Y    ;
00199  C16A  E6 22            INC $22        ;RIAGGIUNGE 4 NUMERI
00200  C16C  E6 22            INC $22        ;AL PUNTATORE (BYTE BASSO)
00201  C16E  E6 22            INC $22        ;
00202  C170  E6 22            INC $22        ;
00203  C172  C8               INY            ;INCREMETA CONTATORE
00204  C173  C4 FC            CPY LEN        ;PER PRELEVARE LE ALTRE LETTERE
00205  C175  D0 E7            BNE TRASLA     ;CONFRONTA LE LETTERE SPOSTATE C
ON
00206  C177  C6 22            DEC $22        ;LA LUNGHEZZA DEL NOME
00207  C179  C6 22            DEC $22        ;POI RIDECREMENTA IL PUNTATORE D
I 4
00208  C17B  C6 22            DEC $22        ;PER POTERCI AGGANCIARE
00209  C17D  C6 22            DEC $22        ;IL SUFFISSO ",S,W"
00210  C17F  A9 2C            LDA #$2C       ;',' = ASCII $2C
00211  C181  91 22            STA ($22),Y    ;
00212  C183  C8               INY            ;
00213  C184  A9 53            LDA #83        ;'S' = ASCII #83
00214  C186  91 22            STA ($22),Y    ;
00215  C188  C8               INY            ;
00216  C189  A9 2C            LDA #$2C       ;','
00217  C18B  91 22            STA ($22),Y    ;
00218  C18D  C8               INY            ;
00219  C18E  A9 57            LDA #87        ;'W' = ASCII #87
00220  C190  91 22            STA ($22),Y
```

```
TYPE/64......PAGE 0005

LINE# LOC   CODE         LINE


00221  C192  18                   CLC          ;AGGIUNGE 4 ALLA NUOVA
00222  C193  A5 FC                LDA LEN      ;LUNGHEZZA DEL FILE
00223  C195  69 04                ADC #4
00224  C197  A6 22                LDX $22      ;CARICA IN X E Y
00225  C199  A4 23                LDY $23      ;I NUOVI PUNTATORI DEL NOME
00226  C19B  60                   RTS          ;
00227  C19C              ;-------------------------------
00228  C19C  A0 10       PAUSA    LDY #$10     ;ESEGUE UNA PAUSA
00229  C19E  A2 00       LP2      LDX #$00     ;COME IL BASIC:
00230  C1A0  CA          LP1      DEX          ;FORY=16TO0STEP-1
00231  C1A1  D0 FD                BNE LP1      ;FORX=255TO0STEP-1
00232  C1A3  88                   DEY          ;NEXTX
00233  C1A4  D0 F8                BNE LP2      ;NEXTY
00234  C1A6  60                   RTS          ;
00235  C1A7              ;-------------------------------
00236  C1A7                       .END

ERRORS = 00000

SYMBOL TABLE

SYMBOL VALUE
 ADD      C15A    ALTRAS   C0C4    ANCORA   C062    BASIC    A474
 CHKIN    FFC6    CHKOUT   FFC9    CHRGE2   0079    CHRGET   0073
 CHRIN    FFCF    CHROUT   FFD2    CLALL    FFE7    CLOSE    FFC3
 CLRCHN   FFCC    COPYST   00F9    DATO     00F8    DEVICE   00BA
 FINE     C038    FLEN     00B7    FNAME    00BB    FNUM     00B8
 FOUND    C147    GETSTR   AD9E    INIT     C08C    KEYBRD   00C5
 LEGGE    C0E0    LEN      00FC    LP1      C1A0    LP2      C19E
 LP3      C11F    MODE     00F7    MODE1    C031    NOFINE   C12F
 NOTFND   C03E    NUMERR   00FA    OK       C0B1    OPEN     F34A
 OPEN15   C07A    OPEN8    C0B2    OPEN9    C0FC    PAUSA    C19C
 RETURN   C116    SADDR    00B9    SCRIVE   C0F0    ST       0090
 STRING   C0A1    STRPUN   B6A3    TORNA    C0DF    TORNA1   C143
 TORNA2   C146    TRASF    C01E    TRASLA   C15E

END OF ASSEMBLY
```

di Sergio Santostasi

6499 + TELEFONO, FELICE UNIONE

Un'applicazione utile dell'obsoleto adattatore telematico della Commodore

Il vecchio 6499, forse abbandonato in un cassetto, può essere utilizzato per scrivere programmi originali

In passato molte riviste del settore informatico tra cui anche *Commodore* del febbraio 1985 (edita dalla **Systems**), hanno presentato progetti per realizzare un **combinatore telefonico** per il Commodore 64.

Un semplice circuito elettronico, posto sul connettore della **User Port** e collegato alla spina del telefono, permetteva di comporre il numero dell'abbonato sulla tastiera del computer. Il software di comando gestiva, inoltre, una completa agenda telefonica.

E' noto, però, che numerosi appassionati di computer hanno poca dimestichezza con il saldatore o con circuiti elettronici.

Quasi certamente anche i più esperti devono ammettere di tirare un sospiro di sollievo quando, collegato il proprio computer a qualche dispositivo autocostruito, vedono comparire all'accensione il prompt iniziale. Nel caso del combinatore computerizzato, oltre alle eventuali difficoltà di ordine pratico, si aggiunge il **divieto di manomissione** degli impianti telefonici sancito dall'articolo 16 del regolamento di servizio **SIP**.

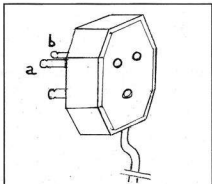
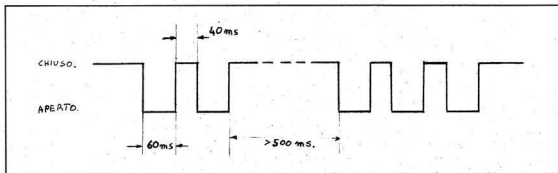
Molti hanno dunque rinunciato a realizzare un dispositivo di tale genere, ma con il breve programma proposto in queste pagine, i possessori dell'adattatore telematico 6499, omologato dalla SIP, possono realizzare, senza alcuna modifica o circuito esterno, un efficientissimo combinatore telefonico.

I lettori potranno poi ampliare a piacimento il programma e, sfruttando la capacità di elaborazione del computer, realizzare, ad esempio, un'agenda telefonica, la ripetizione automatica del numero o effettuare il calcolo dei costi di esercizio del telefono in tempo reale.

Il combinatore telefonico

Per realizzare il combinatore telefonico computerizzato occorre far riferimento al funzio-

Avvertenze

I collegamenti descritti in queste pagine devono essere realizzati con la massima attenzione possibile; in ogni caso è consigliabile la verifica del funzionamento prima di effettuare collegamenti con la linea telefonica.
La Systems Editoriale e l'autore dell'articolo pubblicato, pertanto, declinano ogni responsabilità da danni che dovessero eventualmente verificarsi, anche a causa di er-

L'articolo sarà apprezzato da chi intende utilizzare un'agenda telefonica collegata con la SIP in tempo reale

namento del disco combinatore meccanico e, brevemente, dell'apparecchio telefonico.

Vediamo, in particolare, quali operazioni sono effettuate da parte dell'utente in fase di chiamata.

L'abbonato chiamante solleva il microtelefono (chiamato usualmente "cornetta") ed il contatto di gancio chiude ad anello i fili della linea provenienti da una sorgente di alimentazione posta nella centrale telefonica.

La presenza di una corrente di maglia è quindi interpretata, dalla centrale automatica SIP, come una "richiesta di servizio"; se questa può essere accolta (cioè, quasi sempre) viene inviato al ricevitore dell'apparecchio un segnale, detto tono di centrale.

A questo punto l'utente può iniziare la selezione impostando, sul disco combinatore, la cifra voluta.

Durante la fase di carica del disco, un commutatore esclude il circuito di conversazione per evitare di udire, nel ricevitore, i rumori generati da correnti indotte.

Nella fase di ritorno a riposo, un opportuno albero a camme comanda l'apertura e la chiusura temporizzata del loop di corrente apparecchio - centrale telefonica per un numero N + 2 volte, dove N è la cifra selezionata (se N = 0 vengono generati N + 2 = 12 interruzioni).

Bit 5	Bit 6	**"Effetto" sulla linea telefonica**
0	0	Linea connessa direttamente al telefono.
0	1	Linea sconnessa dal telefono (circuito aperto)
1	0	Linea chiusa su un circuito di impedenza finita.
1	1	Linea chiusa su un corto circuito.

Significato dei bit 5 e 6 della locazione 56834

Gli ultimi due impulsi generati dal commutatore azionato dall'albero a camme, non vengono, però, inviati in linea, ma servono unicamente per garantire una pausa, tra una cifra e la successiva, maggiore di **500 ms**, qualunque sia la cifra selezionata e la velocità dell'utente.

Le specifiche sulla temporizzazione degli impulsi che permettono un corretto funzionamento dei selettori automatici di centrale sono le seguenti:

60 ms di apertura.
40 ms di chiusura.
500 ms pausa intercifra.

La tolleranza di questi valori è abbastanza elevata, dell'ordine del 15%.

In figura è mostrato, a titolo di esempio, l'andamento temporale della selezione delle cifre 2 e 3.

❑❒

L'adattatore telematico

Acceso il Commodore 64, con l'adattatore telematico già inserito, e tornati con **F8** al Basic, proviamo a leggere il contenuto della locazione di memoria 56834: il comando...

Print Peek (56834)

...restituirà il valore decimale **151**.

Normalmente, dunque, i bit **5** e **6** della locazione sono posti a zero.

L'attivazione di questi bit provoca la commutazione di due microrelais interni all'adattatore telematico.

Per verificare ciò, sconnesso l'adattatore dalla linea telefonica, con i comandi...

Poke 56834, 32

...e...

Poke 56834, 64

...potrete udire la commutazione del primo e del secondo relais.

Manipolando i due bit è possibile ottenere, come mostrato in tabella, 4 diverse combinazio-

ni, a ciascuna delle quali corrisponde una diversa configurazione sugli spinotti contrassegnati con **a** e **b** della spina passante SIP dell'adattatore telematico.

Esaminiamo in dettaglio le 4 combinazioni:

1) Poke **PT, 151** (oppure Poke PT, 0). E' il valore di default per il quale l'apparecchio telefonico è connesso alla linea e si pone in attesa di chiamate. E' possibile utilizzare il disco combinatore o il tastierino del telefono per effetture manualmente una chiamata.

2) Poke **PT, 32**. La linea telefonica viene interrotta, ovvero ai suoi capi vi è una impedenza infinita.
Con questa configurazione si realizza l'interruzione di 60 ms della linea durante la generazione degli impulsi decadici.

3) Poke **PT, 64**. Sulla linea telefonica è posta una impedenza pari a quella del circuito di conversazione ottenuta sollevando il microtelefono. Ci si pone in questa configurazione per attendere il tono di centrale.

4) Poke **PT, 96**. I due terminali della linea telefonica vengono posti in corto circuito. Questa configurazione realizza sia la chiusura di 40 ms in ogni impulso che la pausa intercifra di 600 ms.

Analizzato il comportamento sulla linea telefonica dei microrelais dell'adattatore telematico

I bit 5 e 6 della locazione 56834 sono responsabili dei vari collegamenti

```
1 REM ***************************
2 REM *                         *
3 REM * COMBINATORE TELEFONICO *
4 REM *                         *
5 REM *    C/64 & A.T. 6499   - *
6 REM *                         *
7 REM *  BY SANTOSTASI SERGIO   *
8 REM *                         *
9 REM ***************************
10 PT-56834
20 N$ - "080-1234567":REM NUMERO TELEFONICO
22 NB$-"":REM CIFRA SELEZIONATA
23 N-0:    REM IMPULSI DI SELEZIONE
24 T-0:    REM CONTATORE PER PAUSE
25 C-0:    REM CONTATORE CIFRE
26 I-0:    REM CONTATORE IMPULSI
30 PRINT CHR$(147)
40 POKEPT,32: FOR T-1 TO 1800:NEXT:REM RIAPPENDE PER LIBERARE LA LINEA
50 POKEPT,64:FOR T-1 TO 2800:NEXT:REM PRENDE LA LINEA
60 FOR C-1 TO LEN(N$):REM PER TUTTE LE CIFRE CHE COMPONGONO IL NUMERO
70 NB$-MID$(N$,C,1):N-VAL(NB$):REM SELEZIONA UNA SOLA CIFRA
80 PRINT NB$;
90 IF NB$="-" THEN GOSUB 230:GOTO180:REM SE LA CIFRA E' "-" PAUSA
100 IF N-0 THEN N-10:REM SE LA CIFRA E' 0 FARA' 10 IMPULSI
110 POKEPT,96:GOSUB 230:REM PAUSA INTERCIFRA
120 FOR I-1 TO N:REM TANTI IMPULSI QUANTI CORRISPONDONO ALLA CIFRA SELEZIONATA
130 POKEPT,32:T-TI+3.5:REM APRE PER 60 MS
140 IF TI<T THEN 140
150 POKEPT,96:T-TI+2.3:REM CHIUDE PER 40 MS
160 IF TI<T THEN 160
170 NEXT I:REM PROSSIMO IMPULSO
180 NEXT C:REM PROSSIMA CIFRA
190 POKEPT,64:REM FINE DELLA SELEZIONE
200 FOR T-1 TO 700:NEXT
210 POKEPT,0:REM RICOLLEGA IL TELEFONO
220 END
225 REM GENERA LA PAUSA INTERCIFRA
230 T-TI+42
240 IF TI<T THEN 240
250 RETURN: END

READY.
```

Il 6499, con i suoi "miseri" 300 baud, è in grado di offrire un servizio telematico molto costoso; riciclatelo in altre applicazioni

passiamo alla simulazione del funzionamento del disco combinatore e del gancio del telefono mediante il breve programma pubblicato.

❑ ❒

Il programma

Assegnato a **PT** il numero della locazione 56834, e ad **N$** la stringa del numero telefonico da comporre, vengono generati due cicli di attesa: durante il primo (riga 40) viene interrotta la linea e nel secondo (riga 50) si attende il tono di centrale. L'interruzione di linea si rende necessaria per concludere eventuali chiamate precedenti.

Seguono due cicli For... Next: nel più esterno (righe 60 - 180) viene selezionata una cifra per volta del numero telefonico. Il ciclo più interno (righe 120 - 160) provvede poi a generare una interruzione di 60 ms e una chiusura di 40 ms sulla linea telefonica, tante volte quante corrispondono alla cifra selezionata.

La riga 110, posta tra i due cicli, rimanda alla subroutine di pausa intercifra di 600 ms (riga 230). La stessa subroutine è chiamata quando, in riga 90, viene rilevato il carattere meno (-) usato per indicare la pausa forzata tra prefisso e numero dell'abbonato.

Al termine della selezione (riga 190) si porta sulla linea una impedenza pari a quella del circuito di conversazione con il microtelefono sollevato.

Infine, dopo una piccola pausa introdotta per escludere forti rumori nel ricevitore creati dalle correnti indotte nei relais, viene ricollegato il telefono (righe 200 - 210).

A questo punto saranno udibili i segnali di chiamata (o di occupato) provenienti dalla centrale telefoniica.

Si osservi che il telefono risulta sconnesso dalla linea dal primo ciclo di attesa (riga 40) fino al termine del programma (riga 210). Questo rende possibile la combinazione telefonica automatica anche con il microtelefono abbassato: si potrà infatti sollevare il microtelefono in un qualsiasi momento prima della fine della combinazione e attendere che essa si concluda. Questa particolarità diviene utile allorchè il telefono sia posto non molto vicino alla tastiera del computer.

di Alessandro de Simone

DUE EQUAZIONI CON TRE VESTITI

Chi desidera affrontare lo studio di un nuovo linguaggio (il Turbo Pascal, nel caso specifico) spesso si spaventa di fronte alla corposità dei sacri testi; noi, allora, proponiamo...

Molte volte abbiamo sentito una frase del genere: *"A che serve studiare un nuovo linguaggio quando il Basic, che ho imparato con tanta fatica, mi offre gli stessi risultati?"*

Altrettanto spesso, però, queste parole ricordano un po' la vecchia storia della volpe e dell'uva.

Qualche tempo fa, in effetti, era consentita una certa "ignoranza" in materia di linguaggi; oggi, invece, non è assolutamente possibile accontentarsi di un solo linguaggio, se non altro perchè, a scuola, è stato ufficializzato lo studio del **Turbo Pascal** (della Borland) che gira sui computer **Ms-Dos**, standard *suggerito* dal Ministero della Pubblica Istruzione.

Chi, però, ha iniziato con il Basic, può avere difficoltà con le imposizioni di un nuovo linguaggio e rischia, di conseguenza, di entrare ben presto in conflitto con nuove regole sintattiche.

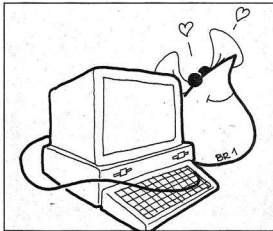
Da questo numero, pertanto, pubblicheremo alcuni listati che, grazie alla loro elementare struttura, possono essere addirittura esaminati senza esser necessariamente digitati; la loro brevità, infatti, non solo consentirà un agevole raffronto tra i linguaggi, ma svolgerà la funzione di evidenziare, fascicolo dopo fascicolo, somiglianze e differenze.

Si descriveranno routines di impiego universale che saranno molto utili, soprattutto a scuola, per risolvere una par-

I linguaggi usati

I programmi pubblicati in queste pagine sono stati scritti con una vecchia versione di **Turbo Pascal** e con una ancora più vetusta versione di Gw-Basic. L'AmigaBasic del terzo programma è siglato 1.3.

Inutile dire che i listati sono perfettamente compatibili con versioni più avanzate dei tre linguaggi dal momento che c'è compatibilità verso il basso. Il motivo della scelta, apparentemente obsoleta, è dovuto alla necessità di venire incontro anche a quella fascia di utenti (e di scuole, nel caso specifico) che per una serie di motivi non si sono potuti procurare versioni aggiornate dei vari linguaggi.

Gli argomenti affrontati, volutamente semplici e banali, non avrebbero comunque subito variazioni di rilievo se sviluppati con versioni più recenti.

te dei problemi assegnati durante le lezioni di informatica.

La routine proposta

I tre programmi pubblicati risolvono un classico problema matematico: la soluzione di un sistema di due equazioni in due incognite.

L'argomento viene affrontato nelle scuole superiori, ma ricordiamo brevemente di che si tratta.

Sia dato il sistema di equazioni...

$a_1 x + b_1 y = c_1$

$a_2 x + b_2 y = c_2$

Bisogna determinare i valori delle due variabili X ed Y tali che, sostituite nel sistema, consentano di soddisfare contemporaneamente le due equazioni.

Ad esempio, il sistema...

$2x + 3y = 4$

$5x + 6y = 7$

...fornisce come risultati:

$X = -1; Y = 2$

Infatti, sostituendo, si ha:

$2 * (-1) + 3 * (2) = 4$

$5 * (-1) + 6 * (2) = 7$

Non tutti i sistemi sono risolvibili. Ad esempio il sistema...

$2x + 3y = 4$

$2x + 3y = 4$

...è indeterminato perchè è possibile trovare una coppia infinita di valori di X ed Y che sono in grado di soddisfare le equazioni.

Allo stesso modo il sistema...

$2x + 3y = 4$

$2x + 3y = 5$

...viene definito impossibile perchè non esistono due valori X ed Y tali da soddisfare il sistema.

Un'equazione è **possibile**, **impossibile** o **indeterminata** a seconda del valore del determinante del sistema che viene definito, come gli altri due determinanti, dalla differenza di due prodotti, ed esattamente:

Determinante del sistema = $a_1 * b_2 - a_2 * b_1$

Determinante di X = $c_1 * b_2 - c_2 * b_1$

Determinante di Y = $a_1 * c_2 - a_2 * c_1$

Al lettore più sveglio, ovviamente, il compito di individuare, nei tre programmi, i tre casi particolari; oppure di rispolverare i libri di scuola...

C'è Gw-Basic e AmigaBasic

La differenza che balza all'occhio, tra un listato scritto in AmigaBasic ed un interprete più tradizionale, è rappresentata senz'altro dalla mancanza dei numeri di linea.

Questo modo di scriver programmi, tuttavia, non è incompatibile con la complessa struttura di AmigaBasic. Se, infatti, digitate su Amiga il listato scritto in Gw-Basic, noterete che questo viene accettato e che, soprattutto, gira(!).

Amiga, infatti, non tiene conto dei numeri di linea (semplicemente li ignora, a meno che non sia presente un **Goto** o un **Gosub**) e tale particolarità consente di caricare programmi che richiedono, invece, la presenza del numero di linea. Il contrario, ovviamente, non può verificarsi (cioè non potete digitare il listato AmigaBasic su un computer Ms- Dos dotato di Gw-Basic).

La differenza tra le due versioni potrebbe addirittura essere meno accentuata dal momento che la subroutine **Leggi:** poteva restare (come nella versione Gw-Basic) in cima al programma, al posto delle righe 110 - 150 e preceduta da un banale **Goto**.

La struttura **If... Then... Else**, sfruttata solo nella versione AmigaBasic, poteva essere usata anche in Gw-Basic; al lettore, ovviamente, è affidato il compito di tradurre il doppio If.. Then (righe 210 - 220) in un più "elegante" If... Then... Else.

La sintassi della funzione **FN** esula dallo scopo del presente articolo; una sbirciatina al manuale AmigaBasic (o Gw-Basic, a seconda dei casi) consentirà di colmare la lacuna.

I più fortunati possessori di Quickbasic (o interpreti similari, come il **Turbo Basic** Borland) potranno divertirsi ad adattare la versione AmigaBasic al proprio interprete / compilatore.

C'è Gw-Basic e Turbo Pascal

Il Turbo Pascal, leggendario pacchetto della **Borland**, è un linguaggio compilatore che, come tale, richiede una certa pignoleria in fase di strutturazione di un programma.

Ai neo-utenti, infatti, può sembrare notevolmente scomodo, tra gli altri, l'obbligo di definire una variabile prima che venga usata. In Basic, come si sa, è invece possibile definire una qualsiasi variabile in un punto qualunque del programma.

Il modo di operare di un compilatore (argomento che approfondiremo un po' per volta nei prossimi fascicoli) è fondamentalmente diverso da quello di un linguaggio interprete.

Mentre quest'ultimo, infatti, sposta blocchi di memoria durante l'esecuzione di un programma (operazione che, però, richiede un notevole tempo di elaborazione), un compilatore non interviene sul programma oggetto; questo, tutt'al più, può agire su file esterni al programma stesso o su segmenti di memoria diversi da quelli occupati dal programma.

Un programma compilato, pertanto, deve contenere, al suo interno, lo spazio necessario per ospitare tutte le variabili che possono essere usate. Analogamente viene riservato spazio alle subroutine, rigorosamente in linguaggio macchina, che possono essere usate nel corso dell'elaborazione.

La definizione preventiva di tutte le funzioni e di tutte le procedure, quindi, facilita il lavoro di indicizzazione e di indirizzamento che, in fase di elaborazione, consente di sviluppare velocità impensabili se paragonate con il procedimento di un qualsiasi interprete.

GoTo

Nel linguaggio interprete non è possibile inserire subroutines dove capita. In caso contrario, come questo, verrebbero elaborate in momenti inopportuni. Come vedremo prossimamente, è bene allocare le subroutines in "testa" al programma. Per fare in modo che non siano elaborate non appena si impartisce il Run, è sufficiente provocare un "salto" alla riga voluta mediante un banale GoTo.

Gw-Basic

La versione Gw-Basic è facilmente comprensibile anche dai lettori alle prime armi. Il "flusso" del programma scorre, istruzione dopo istruzione, attraverso le righe numerate in successione.
Eventuali deviazioni dal percorso obbligato possono essere effettuate mediante le istruzioni **GoTo**, **GoSub** e **Fn**.

```
100 REM programma "SISTEMA"
105 GOTO 160
106 :
107 REM "Procedura" LEGGI
110 PRINT "(Separare con una virgola i tre valori)"
120 PRINT "Digita i coeffic. della prima equazione";
130 INPUT A1, B1, C1
140 PRINT "Digita i coeff. della seconda equazione";
150 INPUT A2, B2, C2: RETURN: REM Fine "Procedura"
151 :
160 DEF FN K (M, N, P, Q) = M*Q - P*N: REM "Function"
170 CLS: PRINT "Sistema di equazioni": GOSUB 110
180 DS = FN K(A1, B1, A2, B2)
190 DX = FN K(C1, B1, C2, B2)
200 DY = FN K(A1, C1, A2, C2)
210 IF DS = 0 AND DX = 0 THEN PRINT "Equazione indeterminata": END
220 IF DS = 0 THEN PRINT "Equazione impossibile": END
230 PRINT: PRINT "Ecco i risultati:": PRINT
240 X = DX/DS: Y= DY/DS
250 PRINT "x = " X " y = "Y;
260 PRINT: PRINT: PRINT "Infatti:": PRINT
270 PRINT A1 "* (" X ")";: IF B1 >= 0 THEN PRINT " +";
280 PRINT B1 "* (" Y ") =";: IF C1 >=0 THEN PRINT " +";
290 PRINT C1: PRINT
300 PRINT A2 "* (" X ")";: IF B2 >= 0 THEN PRINT " +";
310 PRINT B2 "* (" Y ") =";: IF C2 >=0 THEN PRINT " +";
320 PRINT C2: END
```

Def Fn...

Il linguaggio Basic, in tutte le sue versioni, consente di determinare il valore di funzioni matematiche, anche se complesse. La sintassi è del tutto simile a quella riscontrabile nei linguaggi compilatori, come il Turbo Pascal. Le variabili "di comodo" (M, N, P, Q) vengono qui utilizzate, dall'interprete, per sostituirvi i valori delle variabili "passate" dal comando DX = FN K (e similari).

Funzioni

Le tre righe di programma consentono di calcolare i tre detrminanti richiesti. Nel primo caso (riga 180), l'interprete salta ad elaborare la riga 160 dal momento che la 180 "invoca" la funzione di nome "K". Non appena FN K viene intercettata, alle variabili di comodo (M, N, P, Q) vengono attribuiti, nello stesso ordine, i valori contenuti nelle variabili A1, B1, A2, B2. Subito dopo viene eseguita la differenza tra i due prodotti che, invece della simbolica...

M*Q - P*N

...si esplicita in...

A1 * B2 - A2 * B1

...grazie, appunto, alla sostituzione prima effettuata. In seguito il programma riprende con la riga successiva (cioè la 190).

Print

Per visualizzare l'Output, su schermo, vengono usati alcuni trucchetti che consentono di realizzare spaziature o allineamenti. Al lettore il compito di studiare il metodo usato nell'ultima (oltremodo banale) parte del listato

Le variabili in Turbo Pascal

Come accennato nel testo, è obbligatorio definire una variabile prima di usarla. Tale accorgimento è necessario anche scrivendo subroutines e funzioni, come appunto si può notare in questo stesso listato. Si noti, inoltre, il simbolo di punto e virgola (;) che svolge funzioni analoghe a quelle del doppio punto (:) del Basic.

Procedure

Questa procedura, in effetti, non era indispensabile per il corretto svolgimento dell'elaborazione; è stata tuttavia inserita per far comprendere, al principiante, le modalità di inserimento di una procedura che può essere "invocata" in un punto qualsiasi del programma (vedi, appunto, il comando **leggi;** presente poco più avanti).

```
program SISTEMA;
var a1, b1, c1, a2, b2, c2, ds, dx, dy, x, y: real;
procedure leggi;
begin
     writeln ('(Separare con uno spazio i tre valori)');
     write ('Digita i coeffic. della prima equazione ');
     readln (a1, b1, c1);
     write ('Digita i coeff. della seconda equazione ');
     readln (a2, b2, c2);
end;

function deter (m, n, p, q: real): real;
var det: real; begin det:= m*q - p*n; deter:= det end;

begin
    clrscr; writeln ('Sistema di equazioni');
    leggi;
ds:= deter (a1, b1, a2, b2);
dx:= deter (c1, b1, c2, b2);
dy:= deter (a1, c1, a2, c2);
    if ds = 0 then
        begin
            if dx=0 then writeln ('Equazione indeterminata')
                else writeln ('Equazione impossibile')
        end
    else
        begin
            writeln; writeln; writeln ('Ecco i risultati:');
            x:=dx/ds; y:=dy/ds;
            writeln ('x = ',x:0:2,'; y = ',y:0:2);
            writeln; writeln ('Infatti:'); writeln;
            write (a1:0:2,' * (',X:0:2,') ');
            if b1 >= 0 then write ('+');
                write (b1:0:2,' * (',Y:0:2,') = ');
            if c1 >= 0 then write ('+');
                write (c1:0:2); writeln; writeln;
                write (a2:0:2,' * (',X:0:2,') ');
            if b2 >= 0 then write ('+');
                write (b2:0:2,' * (',Y:0:2,') = ');
            if c2 >= 0 then write ('+');
                write (c2:0:2);
        end

end.
```

Begin... end

La cosiddetta **ricorsività** del Turbo Pascal obbliga ad usare "blocchi" razionali di istruzioni, delimitati da **Begin** (in cima) e da **End** (alla fine). Chi studia il T. Pascal per la prima volta rimane sconcertato dalla presenza di un (apparentemente) infinito numero di blocchi; la pratica, tuttavia, consentirà di individuare immediatamente blocchi omogenei e, soprattutto, costringerà l'utente a scrivere i programmi con molta attenzione, evitando i tipici errori dovuti alla frettolosità consentita, ad esempio, dal Basic.

Indentazione

Con questo termine si intende la spaziatura, *eventualmente* lasciata all'inizio di ogni rigo di programma, che ha lo scopo di evidenziare con maggiore efficacia i blocchi logici del listato stesso. L'indentazione non è "vista" dal compilatore che funziona egregiamente anche in assenza totale di indentazione (vedi, a tal proposito, il blocco relativo alla funzione **Deter**, che contiene **Begin** ed **End** sulla stessa riga).

Tra breve, su queste pagine, parleremo *sistematicamente* di...

MONDODOS:

Linguaggio C
Linguaggio Turbo Pascal
Linguaggio Quick Basic
Linguaggio Assembly 8088 / 80486
Sistema Operativo Ms - Dos
Hardware: schede e accessori

PIANETA AMIGA:

Linguaggio C (versione Amiga)
I Basic (interpretati e compilati)
Linguaggio Assembly 68000
Sistema Operativo Amiga Dos
Hardware: schede e accessori

La nostra testata vanta, da quasi un decennio, una notevole esperienza nel campo della didattica; moltissimi sono gli utenti del personal computer che devono le loro conoscenze all'impostazione della rivista che hai tra le mani.
Per venire incontro alle esigenze dei nuovi utenti, senza trascurare quelle dei suoi affezionati lettori (che, nel frattempo, si sono sensibilmente "evoluti") è stata presa la decisione di approfondire argomenti adeguati alle macchine degli anni '90.

Funzioni

Il modo di definire una funzione non cambia rispetto alle modalità richieste da versioni Basic più modeste. Anche con **AmigaBasic** è sufficiente invocare la funzione, per attivarla, passando i vari parametri richiesti. Questi devono esser "congrui" (come qualità e quantità) con quelli presenti nella riga contenete l'istruzione **Def Fn.**

If... Then... Else... End If

La forma sintattica If.. Then, classica istruzione di **salto condizionato**, deve essere ben compresa per evitare segnalazioni di errori. Nel nostro caso sono presenti due **nidificazioni** di If.. Then. Ad ogni **If** corrisponde, opportunamente *indentato* , il suo limite "logico" (cioè: **End If**). Analogamente l'istruzione **Else** deve essere posizionata in modo opportuno per evitare che vengano elaborate, erroneamente, altre istruzioni. Si noti, anche, la provvidenziale presenza di due **End** all'interno del blocco.

```
REM programma "SISTEMA"
DEF FN K (M, N, P, Q) = M*Q - P*N: REM "Function"
CLS: PRINT "Sistema di equazioni": GOSUB Leggi:
DS = FN K (A1, B1, A2, B2)
DX = FN K (C1, B1, C2, B2)
DY = FN K (A1, C1, A2, C2)
    IF DS = 0 THEN
        IF DX = 0 THEN
            PRINT "Equazione indeterminata": END
            ELSE: PRINT "Equazione impossibile": END
        END IF
    END IF
PRINT "Ecco i risultati:": PRINT
X = DX / DS: Y = DY / DS
PRINT "x = " X " y = "Y;
PRINT: PRINT: PRINT "Infatti:": PRINT
PRINT A1 "* (" X ")";: IF B1 >= 0 THEN PRINT " +";
PRINT B1 "* (" Y ") =";: IF C1 >=0 THEN PRINT " +";
PRINT C1: PRINT
PRINT A2 "* (" X ")";: IF B2 >= 0 THEN PRINT " +";
PRINT B2 "* (" Y ") =";: IF C2 >=0 THEN PRINT " +";
PRINT C2: END
    Leggi:
        PRINT "(Separare con una virgola i tre valori)"
        PRINT "Digita i coeffic. della prima equazione";
        INPUT A1, B1, C1
        PRINT "Digita i coeff. della seconda equazione";
        INPUT A2, B2, C2:
    RETURN: REM Fine "Procedura"
END
```

Anche con AmigaBasic, come in **Turbo Pascal** (e in **QuickBasic** e **Turbo Basic**) è possibile ricorrere all'**indentazione**, miracolosa potenzialità ben apprezzata da chi è costretto spesso a "riprendere" precedenti versioni di programmi, allo scopo di modificarli. La comoda lettura di un programma, infatti, è basilare per apportare le correzioni del caso dopo aver individuato, agevolmente, il segmento di listato che interessa.

Subroutines

Le subroutines di AmigaBasic assomigliano moltissimo alle **procedure** del Turbo Pascal; è infatti sufficiente "invocarle" con il loro nome (vedi, appunto, **GoSub Leggi** all'inizio) per attivarle. La sintassi dell'interprete consente, in realtà, un uso più sofisticato, tra cui il "passaggio" di parametri dal programma principale. Anche in questo caso si poteva fare a meno di una subroutine; si è preferito inserirla, tuttavia, per meglio valutare le somiglianze tra AmigaBasic e T. Pascal.

di Giancarlo Mariani

COME ANIMARE UN CERCHIO

Continua la serie di articoli dedicati al linguaggio C, che sta riscuotendo notevole interesse tra i nostri lettori; stavolta "entriamo" nella grafica del nostro potente computer Ms-Dos compatibile

Dopo aver parlato delle quattro operazioni (vedi n. 78) parleremo ora di qualcosa di un po' più complicato (ma non troppo): la **grafica** ed il trattamento dei files **sequenziali**.

Il **Turbo-C** (marca Borland, ma anche di altre s/w house) dispone di potentissime istruzioni dedicate alla grafica che consentono di disegnare, cancellare, muovere, colorare e modificare qualsiasi figura sul video.

Inoltre è possibile creare **finestre**, memorizzarle in variabili, disegnare istogrammi tridimensionali, e tante altre cose che fanno del tool grafico del Turbo-C un prodotto veramente eccellente.

Il programma di cui ci occupiamo stavolta è una semplicissima applicazione grafica, che comprende un altrettanto semplice gestione di file.

Il programma consente di:

- **Disegnare** sullo schermo un cerchio, con coordinate del centro e raggio a piacere.
- **Spostare** il cerchio stesso ovunque sullo schermo, tramite i tasti cursore.
- **Rivedere**, con la semplice pressione di un tasto, l'intera sequenza di movimenti effettuata, e quindi **animare** il cerchio.
- **Registrare** su file sequenziale sia i parametri del cerchio sia la sequenza di movimenti.
- **Caricare** da file sequenziale i parametri del cerchio e la sequenza di movimenti per osservare nuovamente l'effetto di animazione.

Ovviamente il programma non ha alcun utilizzo pratico, se non quello di far meglio comprendere la gestione della grafica e dei files sequenziali.

Come gira

Una volta digitato e fatto partire, il programma presenta un menu con 4 opzioni, tramite il quale è possibile effettuare altrettante procedure.

1) Movimento con tasti.

Premendo il tasto **1** (senza premere Enter) si entra nella fase di movimento "manuale" del cerchio. Dapprima ne verranno richiesti i parametri (coordinate **X** e **Y** del centro e **Raggio**) da digitare separati da una virgola; quindi andrà premuto Enter.

NB: per non complicare il programma, questo non esegue alcun controllo sui parametri inseriti, quindi

L'angolo del C

Riservato ai principianti

Supponendo che abbiate già installato il compilatore sul vostro computer, le operazioni da compiere sono le seguenti:

1) Posizionatevi nella directory che lo contiene e lanciatelo tramite il comando **TC**.

2) Premere i tasti Alt + E (edit) per accedere alla finestra di edit, e digitare con la massima cura il listato pubblicato sulla rivista.

3) Premere il tasto F2, inserire il nome (eventualmente completo di **Path**) del programma appena digitato per salvarlo su disco.

4) Premere i tasti Alt - C e selezionare l'opzione **Make_Exe_File** per compilare il listato. Al termine della compilazione compare una finestra contenente alcuni dati sull'operazione compiuta. La parola **Success** indicherà una compilazione andata a buon fine, a meno che non siano presenti messaggi di **Warnings**. Bisogna infatti sottolineare che il compilatore C segnala, oltre agli errori veri e propri, anche le avvertenze (Warning), che non sono errori, ma che possono rappresentare una possibile fonte di errore,solitamente di tipo **logico**.

In quest'ultimo caso, gli "inconvenienti" verranno evidenziati in una finestra in basso ed il cursore si posizionerà sul primo di questi. Tramite i tasti freccia in alto e freccia in basso potremo spostarci lungo la lista e, premendo il tasto **F6**, posizionare il cursore sul listato sorgente nel prunto esatto in cui è stato riscontrato l'errore, che potremo controllare e correggere con facilità. Premendo di nuovo F6 si tornerà alla finestra degli errori per selezionare, eventualmente, un altro errore.

Premendo **F1** nella finestra degli errori, verrà visualizzata la finestra di aiuto, relativa all'errore occorso, che contribuirà a chiarire le idee in caso di dubbi.

Bisogna ricordarsi di salvare **sempre** il listato dopo ogni correzione di errore, ed in ogni caso anche durante la digitazione per evitare di dover ridigitare il programma in caso di "inchiodamento" del computer per mancanza di tensione o altro. Il passo 4 va ripetuto sino a che la compilazione va a buon fine: comparsa del messaggio "Success", numero Warnings = 0 e numero Errori = 0. Alcuni messaggi di warnings del tipo *call to function xxxxxx without prototype* possono essere prodotti dalla mancanza di righe **include**.

Il compilatore C produce il file eseguibile (suffisso **.EXE**) direttamente su disco in modo che il programma possa girare anche in assenza del compilatore.

5) A questo punto si possono osservare i risultati premendo i tasti **Ctrl - F9**, sequenza che manda in esecuzione il programma appena compilato.

Se osservate elaborazioni diverse da quelle che aspettavate bisognerà stavolta controllare *non* la **sintassi ma la logica**, perchè sicuramente avrete sbagliato qualcosa. Nella digitazione dei listati bisogna prestare particolare attenzione ad alcuni particolari: il compilatore differenzia le lettere **minuscole** da quelle **maiuscole**. Il listato, comprese le variabili, va digitato esattamente come compare sulla rivista, pena segnalazioni di (in)spiegabili errori. In particolar modo, tutte le **parole chiave** andranno digitate in minuscolo. Inoltre, l'operatore di assegnazione eguale (=), contrariamente a quello che succede in linguaggi come il **Basic**, è diverso da quello di uguaglianza (dopio eguale, cioè ==); si dovrà, quindi, scrivere ad esempio, nel caso di assegnazione (**A = B**) e, in caso di confronto...

if (A==B)

...e non...

if (A=B)

...che in C equivarrebbe ad una istruzione di assegnazione **A = B** con la conseguenza che verrebbe effettuato un confronto del risultato con zero, del tipo...

if (A < > 0)

Tale svista, naturalmente, produce un risultato completamente diverso da ciò che si aspetta, pur se viene segnalato dal compilatore con un Warning del tipo *Possibly incorrect assignment* che, però, non arresta la compilazione... In C l'operatore **diverso da** è composto da un punto esclamativo ed un uguale (**!=**) e **non**, come in altri linguaggi, dai segni minore e maggiore (< >). L'espressione **se A è diverso da B** diventa, in C:

if (A != B)

Le stringhe sono trattate TUTTE come puntatori: non si può assegnare una stringa ad un'altra con un'istruzione del tipo...

Stringa2 = Stringa1

Questa sintassi, in C, significherebbe: **il puntatore Stringa2 punta ora a Stringa1.**

Si otterrebbe, come risultato, non solo il fatto che Stringa1 e Stringa2 punterebbero alla stessa area di memoria (cioè risulterebbero una **variabile unica**), ma anche che il puntatore all'area precedentemente puntata da Stringa2 verrebbe **perso**, non permettendo di recuperare in alcun modo quell'area di memoria.

L'assegnazione tra le stringhe deve invece essere effettuata tramite l'istruzione...

strcpy (Stringa2, Stringa1)

...che copia le due stringhe correttamente, lasciando inalterati i relativi puntatori.

In ogni caso il compilatore C possiede un **help** in linea, disponibile in ogni momento con la pressione del tasto **F1**, grazie al quale si accede all'help generale, che contiene informazioni relative all'editor ed al pacchetto in generale. Premendo **Ctrl + F1** con il cursore posizionato su di una parola chiave, verrà visualizzato l'help ad essa relativo corredato, a volte, di un esempio di programmazione.

Se viene premuto Ctrl + F1 quando il cursore non è posizionato su di una parola chiave, viene proposta la lista di tutte le parole e si potrà scegliere quella della quale si vuole l'help.

Originale è bello

I lettori di C.C.C. sono abituati ad una nostra interpretazione decisamente disinvolta per quanto riguarda la protezione del software (ed i diritti d'autore). Se, però, il videogioco spara e fuggi spesso non vive tanto da meritare il denaro richiesto per l'acquisto del s/w originale, nel caso di un pacchetto professionale le cose cambiano.

Purtroppo numerosi sono i pacchetti applicativi che superano abbondantemente il mezzo milione di lire e lo studente squattrinato (e non solo lui) preferisce arrangiarsi operando con copie "di favore" (scusate l'eufemismo...) avendo a portata di mano qualche fotocopia che spiega, in linea di massima, il principio di funzionamento.

Un w/p, utilizzato prevalentemente per scrivere qualche lettera, non richiede particolari conoscenze e, dopo alcuni tentativi, chiunque è in grado di manovrarlo con sufficiente rapidità.

Nel caso di un linguaggio, però, le cose cambiano, e di molto.

Anche se i moderni compilatori offrono i preziosissimi Help (addirittura in linea) il manuale originale, c'è poco da fare, è decisamente indispensabile. Basta fare un paio di conti per rendersene conto. Se la copia di favore si ottiene a bass(issim)o prezzo (e non ditemi che si può ottenere completamente gratis), almeno un libro sul linguaggio bisogna pur comprarlo: non illudetevi che gli help in linea siano sufficienti. Una volta acquistato il volume (in lingua inglese, per carità!) ci accorgiamo che dal portafogli sono già usciti almeno sei - sette biglietti da 10000; per accorgersi, magari, che il libro, per ovvi motivi, non può riportare tutte le istruzioni per usare correttamente il linguaggio; spesso capita di leggere un laconico messaggio del tipo *"...per maggiori informazioni consigliamo di esaminare il manuale in dotazione del pacchetto..".*

E allora?

Vale la pena tentare di risparmiare poche decine di migliaia di lire per evitare di comprare la bella confezione originale del linguaggio, con tutti i vantaggi offerti (tra cui la possibilità di essere inseriti in una mailing list per ottenere informazioni su futuri sviluppi del linguaggio, upgrade a prezzi ridotti, possibilità di usufruire di eventuali hot line, eccetera).

Il nostro consiglio, quindi, è quello di procurarsi i package originali anche perchè abbiamo intenzione di pubblicare routine applicative che presupporranno una conoscenza abbastanza approfondita dei manuali.

---

bisognerà aver cura di non eccedere i limiti della scheda grafica posseduta. Per una **CGA**, ad esempio, il valore X può variare tra **0** e **639**, mentre Y tra **0** e **199**. Il punto (0, 0) è posizionato, come al solito, in alto a sinistra.

Una volta inseriti i parametri si entrerà in grafica, il cerchio verrà disegnato e sarà possibile muoverlo con i tasti cursore. Anche in questo caso lo spostamento non viene controllato, quindi il cerchio si potrà spostare anche "fuori" dallo schermo.

La fase di spostamento ha termine quando si preme il **tasto Enter**: a questo punto il cerchio viene riportato nella posizione originale e tutti i movimenti effettuati (fino a **10000**!) vengono ripetuti automaticamente.

Alla fine dell'animazione il programma attende la pressione di un tasto prima di tornare al menu principale.

2) Movimento in automatico.

Questa opzione, selezionabile da menu principale tramite il tasto **2**, produrrà lo stesso effetto dell'opzione precedente alla pressione del tasto Enter muovendo il cerchio in accordo con quanto digitato precedentemente tramite l'opzione **1** oppure **4**.

Con la pressione di un tasto si ritorna al menu principale.

3) Salva movimenti su disco.

Con questa opzione (tasto **3** da menu principale) i parametri del cerchio ed i movimenti eventualmente presenti in memoria verranno salvati su di un **file sequenziale**, il cui nome verrà richiesto dal programma.

NB: Se il file esiste già su disco, verrà cancellato e quindi riscritto con i nuovi contenuti. Se il nome contiene caratteri non ammessi, oppure è troppo lungo, o digitate altre inesattezze del genere, il file semplicemente non verrà creato, ma il programma **non** produrrà messaggio di errore.

4) Carica movimenti da disco.

Con la pressione del tasto **4** sarà possibile caricare, da disco, i parametri del cerchio e la sequenza di movimenti precedentemente registrati con l'opzione 3. Il programma chiederà il nome del file; se questo non esiste, oppure il nome contiene caratteri non corretti, il file semplicemente non verrà caricato, però **non** verrà prodotto alcun messaggio di errore.

5) Fine programma.

Si ottiene premendo il tasto **ESC**ape.

NB: Il cerchio ed i movimenti eventualmente presenti in memoria andranno persi!

Il listato

Anche se il programma è molto semplice (è destinato ai principianti, diamine!) una breve descrizione può essere utile a coloro che vogliono approfondire le proprie conoscenze del C.

Vediamo ora in dettaglio le varie parti del listato.

1) **Commenti**. Sono racchiusi tra *barra asterisco* (/*) (inizio) e *asterisco barra* (*/) (fine del commento) e possono venir scritti anche su più linee. Corrisponde alla **REM** del Basic.

2) **Files di include**. Questi vengono inclusi nel listato con la direttiva...

#include < >

...e generalmente contengono la parte di dichiarazione delle procedure, le variabili e le costanti necessarie alle istruzioni del C.

Vediamo di chiarire le idee nel caso del nostro programma.

stdio.h (abbreviazione di **standard i/o**) è la parte di dichiarazione delle istruzioni di tipo scanf, printf, la gestione dei files, ed altre.

conio.h (**con**sole **i/o**) serve per attivare le operazioni di input / output da console, come getch (get character), clrscr (clear screen) ed altre.

stdlib.h (**st**andard **lib**rary) serve per le conversioni, tipo itoa, atoi.

graphics.h definisce le funzioni grafiche.

3) Definizione delle **costanti globali**. Queste verranno "viste" dall'intero programma, comprese le procedure e le funzioni. Nel nostro caso definiscono i codici dei tasti cursore, di Enter e del tasto Escape (ESC).

4) Definizione delle **variabili globali**. Anche queste variabili, come le costanti globali, possono esser prese in considerazione in qualunque parte del programma, con la differenza che possono essere modificate a piacere.

L'**array ch** contiene l'elenco dei tasti premuti, le variabili intere **X**, **Y**, **R**, **OldX**, **OldY** servono per contenere i parametri del cerchio; nell'array **Movim** (dimensionato a 10000), verranno memorizzati i movimenti effettuati, uno per ciascun elemento; infine la variabile conta indica il numero dei movimenti effettuati e, in pratica, è l'indice dell'array Movim.

5) **Prototipi di funzione**. Queste righe, che copiano fedelmente la riga di dichiarazione posta all'inizio di una procedura o funzione, servono ad un duplice scopo: innanzitutto informano il compilatore su quante e quali sono le subroutine e le funzioni che vogliamo scrivere; inoltre consentono di controllare il "passaggio" dei parametri alle funzioni stesse.

Omettendo la dichiarazione dei prototipi di funzione, il programma funziona ugualmente (a parte qualche messaggio di Warning); se, però, passiamo una stringa ad una funzione che accetta parametri interi (o viceversa), oppure dimentichiamo un parametro nel chiamare la funzione, il compilatore non se ne accorgerà, producendo risultati imprevedibili.

I files di include (**.H**) contengono, appunto, questi prototipi, riferiti però non alle procedure sviluppate dall'utente, ma alle funzioni standard del C.

6) A questo punto iniziano le **funzioni e procedure** vere e proprie. Esaminiamole in dettaglio:

Grafica

Attiva lo schermo in modo grafico. L'istruzione **detectgraph** determina automaticamente il tipo di **scheda grafica** presente sul calcolatore, e ne pone i parametri nelle variabili passate, nel nostro caso **g_driver** e **g_mode**. Il simbolo et (**&**) che compare davanti al nome delle variabili significa che viene passato alla funzione NON il **valore** della variabile, ma il suo **indirizzo**. Tale modo di operare consente alla funzione di modificare il valore della variabile passata, cosa che non sarebbe possibile altrimenti.

Initgraph utilizza i parametri ricavati dall'istruzione precedente per mettere effettivamente lo schermo in grafica. L'ultimo parametro (**c:\\tc**) indica la directory, o meglio il percorso da seguire per rintracciare i driver grafici del Turbo - C, che sono forniti assieme al compilatore ed hanno l'estensione **.BGI** (**B**orland **G**raphics **I**nterface).

Il driver per la CGA si chiama **CGA.BGI**, quello per la EGA **EGAVGA.BGI**, quello per la Hercules **HERC.BGI**. La doppia controbarra nel percorso (c:\\tc e non c:\tc) serve perchè il C considera la controbarra posta in una stringa come l'inizio di un carattere di controllo. Per fare in modo che venga effettivamente

Per chi suona la campana

Potrebbe sembrare una contraddizione il nostro consiglio sull'acquisto del s/w originale e sulla contemporanea pubblicazione di articoli (e programmi) che consentono di scrivere un listato a chi dispone di copie pirata dei linguaggi presi in esame.

Alcune note di queste pagine, infatti, spiegano come attivare correttamente la procedura per editare, correggere, compilare e memorizzare un programma; operazioni queste, lo sappiamo benissimo, certamente riportate sul manuale di istruzioni originale che, in teoria, dovrebbe essere in possesso del lettore cui sembriamo rivolgerci. Le informazioni elementari che pubblichiamo, quindi, sono destinate a coloro che, pur essendo entrati in possesso del s/w piratato, non riescono ad utilizzarlo a causa della mancanza di informazioni al riguardo. La caramella (leggi: i brevi programmi delle nostre pagine) che offriamo a questa categoria di lettori per attirarne l'attenzione, dunque, rappresenta un invito a "provare" un linguaggio che, appunto, posseggono, ma non utilizzano.

Una volta sperimentata la straordinaria potenza del compilatore (e non può essere altrimenti), sfatato il mito della difficoltà di comprensione rispetto al Basic (basta un po' di buona volontà), intuita la potenzialità offerta da un linguaggio moderno ed evoluto, il prossimo passo da compiere sarà inevitabilmente quello di recarsi presso il più vicino rivenditore di s/w e procurarsi il prodotto originale.

Per poi incontrarci, puntualmente ogni mese, e sviluppare nuove tecniche di programmazione, alla ricerca perenne della perfezione...

Commodore
COMPUTER
CLUB
La rivista degli utenti di sistemi Commodore

recensioni

CORPORATION

Una multinazionale è alle prese con un progetto pericoloso; urge il vostro intervento

Computer: Amiga inespanso
Gestione: Joystick e tastiera
Tipo: Avventura animata 3D
Softhouse: Core Design

Corporation è basato sulla grafica animata tridimensionale in tempo reale, che ha già decretato il successo di tante riedizioni delle vecchie avventure con solo testo.

Il gioco

La Universal Cybernetics Corporation (UCC) è una multinazionale tanto grossa da fare apparire la IBM e la Exxon come dei bottegai. Il suo successo è dovuto allo sviluppo di robots per usi domestici e commerciali. Voci insistenti dicono che si stanno muovendo verso la manipolazione genetica per creare forme di vita nuova, ibride tra uomo e macchina, con lo scopo di creare delle perfette macchine da guerra. Il governo ha notato alcuni strani omicidi nelle zone dei laboratori di ricerca della UCC ed ha deciso di mandare qualcuno (cioè noi) ad investigare. All'inizio è possibile scegliere tra quattro personaggi, quattro umani e due droidi. Ciascun agente ha una sua abilità e personalità. Poi bisogna scegliere l'equipaggiamento, spendendo con oculatezza i soldi a disposizione in armi, protezione ed apparecchi elettronici di ausilio (computer da ricerca, droghe potenzianti, bombe, mappe, pistole, maschere antigas, eccetera). Ovviamente l'equipaggiamento è da scegliere anche in base al personaggio: ad esempio i droidi non hanno bisogno di sensori all'infrarosso per vedere al buio, come gli umani. Una volta terminata la fase preparatoria, il gioco inizia avviandosi in elicottero ai laboratori di ricerca UCC. Il gioco si snoda in una serie incredibile di puzzle, situazioni curiose ed angoscianti ed estremamente stimolanti sia per quanto riguarda le capacità di riflessione e decisione del giocatore, sia per quanto riguarda la sua abilità col joystick.

La tecnica

Lo schermo principale mostra una visione 3D in tempo reale dagli occhi del personaggio.

Gli sprites sono estremamente rifiniti e animati bene, tenuto conto delle dimensioni e della tecnica implementata.

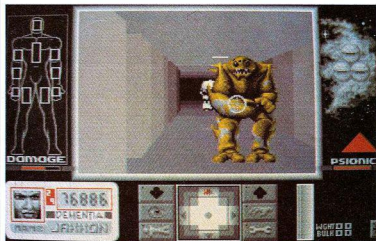
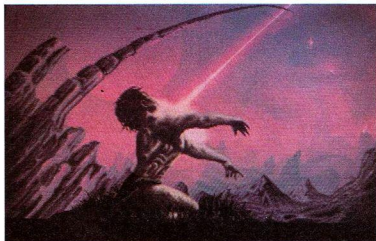
Vi sono ben 16 livelli per un totale di vari kilometri quadrati da esplorare e parecchie centinaia di stanze da visitare. La grafica è molto curata, sia nelle schermate "gestionali", sia nelle animazioni e negli spostamenti in tempo reale.

Il voto

Una bella avventura animata, a metà tra arcade rifinito e avventura "intrigante". 9.

SHADOW OF THE BEAST II

Un gioco che rappresenta il degno "seguito" della famosa, prima avventura

Computer: Amiga inespanso
Gestione: Joystick e tastiera
Tipo: Arcade game
Softhouse: Psygnosis

Se **Barbarian** aveva avuto un seguito, perchè la Psygnosis non avrebbe dovuto produrre anche quello del fortunatissimo **The Beast**?

Il gioco

Nella oscura e distante terra di **Kara-Moon** il perfido **Zelek**, riflette sulla propria posizione: è stato incaricato da **Maletoth**, Signore del Male, di trovare un bambino da forgiare in guerriero. Dopo qualche ricerca lo trova in una capanna e, trasformatosi in uccellaccio, lo rapisce in una impressionante sequenza introduttiva che, probabilmente, occupa buona parte del primo disco di programma!

Il bambino cresce bene, tanto che sconfigge Zelek e fugge dal Beast Lord. Purtroppo deve attraversare **Kara-Moon**, pullulante di avversità omeriche e di mostri che vogliono soltanto fargli la pelle. In effetti si trova anche qualche essere con il quale si può contrattare, ottenendo aiuto in termini di attrezzatura di combattimento o di suggerimenti. Altre volte il contatto con questi abitanti di Kara-Moon apre la strada a nuove piccole avventure nell'avventura, che devono essere affrontate per completarla positivamente.

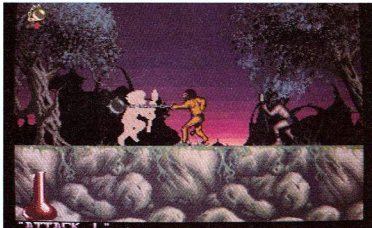
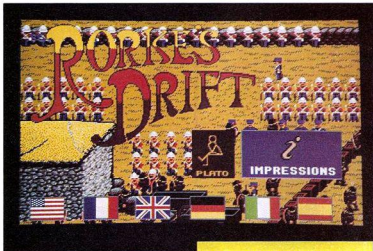
Grafica e suono

Chi ha già giocato con il primo **The Beast** sa che cosa lo attende qui. Gli effetti sonori, l'atmosfera, il tipo di grafica e le animazioni sono chiaramente molto simili, pur se logicamente differenti agli occhi ed all'orecchio, a quelli del predecessore.

Una particolarità è che qui manca lo scrolling parallattico multilivello, ma l'uso più saggio di animazione, colori ed ombre crea una maggiore atmosfera. I mostri appaiono soltanto nel loro ambiente naturale: i pesci nel mare, i pigmei nella foresta, eccetera. I suoni e le musiche sono di alta caratura.

In particolare il motivo del Game Over (tristemente frequente!) presenta un assolo vibrato di chitarra eccezionale, stile Jimi Hendrix!

Il voto

Chi ha già acquistato il primo, comprerà senz'altro il secondo. Agli altri sarà sufficiente vederlo giocare per cinque minuti per decidersi ad acquistarlo! Un 9 è più che meritato.

RORKE'S DRIFT

La maggior parte dei wargames si limitano a simulare sullo schermo i segnalini dei giocatori ed a memorizzare il punteggio. **Rorke's Drift** appartiene invece ad una diversa categoria, chiamata in gergo "miniature wargames", ma sempre di soldatini si tratta...

Impartite ordini ai vostri soldati; in seguito il computer li muoverà nel campo di battaglia

Computer: Amiga inespanso
Tipo: Wargame
Gestione: Mouse
Softhouse: Impressions

Il gioco

L'ambientazione è la battaglia tra 137 soldati britannici e circa 4000 zulu durante il 22 e 23 gennaio 1879, sulla quale fu basata anche la trama del film "Zulu" di Stanley Baker.

L'area in possesso degli inglesi è stata delimitata da sacchi di sabbia ed al suo interno dobbiamo disporre i nostri soldatini per fronteggiare i nemici.

Il gioco è strutturato in due fasi: ordini e combattimento, ambedue ristretti ad un certo tempo limite.

Nella prima fase si decide il lato verso cui è rivolto un soldatino, se deve camminare, correre o presidiare una certa posizione, dove deve caricare la sua arma e dove tornare.

Gli ordini vanno impartiti singolarmente ad ogni uomo, e per fortuna vi è una opzione di ripetizione automatica che consente di assegnare lo stesso compito a più soldati consecutivamente in poco tempo!

Quando si inizia la battaglia, il computer muove gli Zulu e fa eseguire, ai nostri combattenti, gli ordini impartiti, sinchè non li si cambia.

L'orologio, come nella pallacanestro (!), viene azionato durante il combattimento vero e proprio: quando si rientra nel modo "ordini" viene sospeso il conteggio del tempo.

Durante il gioco apposite finestre producono continue informazioni sugli esiti di ogni battaglia. Il giocatore deve provvedere a rifornire di munizioni i soldati, sostituire i caduti in posizioni strategiche ed usare i medicinali per soccorrere i feriti.

La tecnica

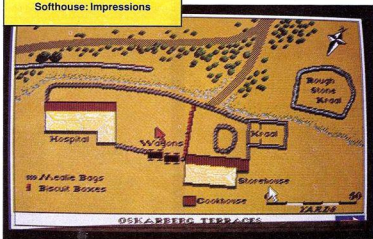
Dimenticate i suoni: non ci sono ed è un vero peccato. La grafica è invece decisamente buona e, pur se piatta, risulta sicuramente più adatta e comprensibile rispetto ad una forzata grafica pseudotridimensionale prospettica.

Il voto

Un buon gioco, originale, tecnicamente valido. 7+.

THE FOOL'S ERRAND

Un gioco che, in effetti, rappresenta una valida collezione di giochi di "riflessione"

Computer: Amiga inespanso
Gestione: Mouse
Tipo: Riflessione
Softhouse: Miles Computing

Jigsaw e Puzzle sono parole che generano ostilità negli amanti dei giochi "tuttogrilletto" e passione sviscerata tra coloro i quali preferiscono riflettere davanti al monitor.

❑❒

Il gioco

Classificato come un'avventura, in effetti questo programma è di genere del tutto particolare'.

L'ispirazione arriva direttamente dal tavolo dei **Tarocchi**! infatti, all'inizio del gioco si può accedere a 21 differenti aree che corrispondono agli **Arcani Maggiori** dei Tarocchi.

Quando si sceglie un'area, dal menu, appare un pezzo di pergamena che illustra una breve storia con i dettagli sul puzzle da risolvere, appunto per consentire al giocatore di sapere che cosa deve fare esattamente.

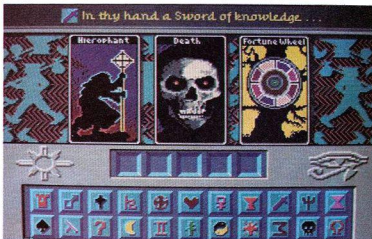
I puzzle sono di tutti i generi e per tutti i gusti: parole incorciate (tipo Scrabble), anagrammi, giochi del 15 ed altro ancora.

Una volta iniziato, ci si ritrova regolarmente impastati sino a notte fonda senza accorgersene!

La tecnica

I suoni sono praticamente inesistenti, e questa è la maggiore e forse unica pecca tecnica del programma. La grafica è effettivamente molto variopinta e fantasiosa, brillante in tutti i punti e nei particolari, tanto che probabilmente spingerà molti giocatori, invischiati dal fascino cervellotico del gioco, ad acquistare un bel monitor per gustare al meglio i particolari.

Il voto

Nel suo genere uno dei migliori programmi. 9-.

COMBAT SIMULATOR

Se lo avessero realizzato per il C/64, il gioco poteva essere valido; ma per l'Amiga...

Computer: Amiga inespanso
Tipo: Tuttogrilletto
Gestione: Joystick, tastiera
Softhouse: Codemasters

Un clone del famosissimo **Commando**, o **Rambo**, se preferite.

Il gioco

Niente bombe contro l'ambasciata, ma il nostro eroe deve semplicemente farsi strada tra quattro fasi nel territorio dei nemici, armato di bombe (lanciate con la barra spaziatrice) e di mitragliatore.

Ogni stadio è suddiviso in due sezioni: la prima parte è identica a Commando, con l'azione vista da sopra, mentre la seconda sezione è vista di lato. Per ottenere armi extra, come ad esempio il fuoco rapido del joystick (se non lo avete già via hardware...) oppure vite extra, si devono raccogliere i vari oggetti disseminati sul percorso.

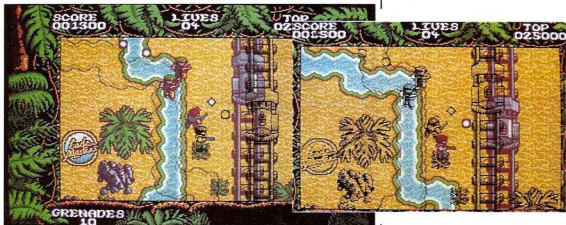
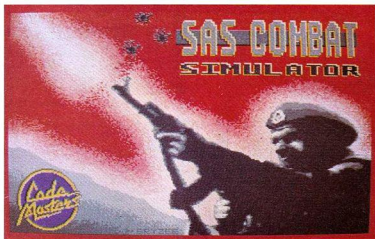
Tali extra vengono però persi quando si muore.

La tecnica

Poca tecnica in questo gioco. Grafica rigorosamente bidimensionale, scrolling ridotto all'essenziale, grafica stilizzata e non immune da difetti di animazione.

Gli scenari, pur nella loro semplicità, sono vari, ma si tratta di una grafica di scena da C/64, o peggio.

Il voto

Solo per i patiti del genere, o forse neanche per loro. 5 1/2.

DUNGEON QUEST

Un'avventura... riposante

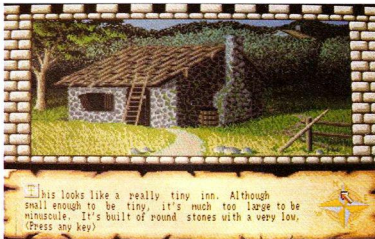
Computer: Amiga inespanso
Gestione: Tastiera
Tipo: Avventura grafica
Softhouse: Gainstar

Sono cambiati i tempi da quando le avventure erano solo testuali. Ora le grandi capacità della memoria e dell'hardware di Amiga consentono alle softhouse migliori di realizzare avventure estremamente rifinite. Non è, purtroppo, il caso di questo programma.

Il gioco

La trama del gioco è quella del leggendario Dungeon Quest, che chiunque abbia un minimo di passione per le avventure dovrebbe conoscere.

L'innovazione è dovuta ad un uso estremo di grafica e suoni.

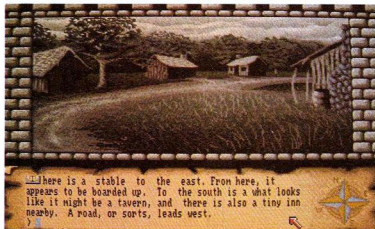
Le schermate sono sempre disegnate con una nitidezza ed una cura dei particolari veramente ammirevoli e gli effetti sonori si adeguano. Ad esempio, se siamo in un sottobosco si odono chiaramente i rumori del vento tra le fresche frasche e delle ali delle garrule che svolazzano attorno.

La trama del gioco è molto piana, senza troppi mostri o situazioni da cronaca nera, come un certo tipo di gusto per l'horror ci ha ultimamente abituato a vedere sui nostri schermi. Vi sono persino pochi personaggi con i quali interagire. In effetti oseremmo dire che si tratta di una avventura riposante...

La tecnica

Il punto debole, anzi debolissimo, del programma è l'implementazione dell'interfaccia utente. Per intenderci con un termine tecnico il cosiddetto "parser, ovvero la parte di programma che legge gli input del giocatore e manovra di conseguenza, è decisamente rudimentale, di certo inferiore alle rifiniture prettamente artistiche già citate.

Capita che sullo schermo si vedano degli oggetti dei quali il programma sembra non conoscere l'esistenza, oppure di impartire ordini ed ottenere risultati del tutto incomprensibili.

Il voto

Eccellente dal punto di vista artistico, buono per la trama, decisamente insufficiente per la tecnica di interazione. 6+.

di Ascanio Orlandini

IL MODEM E' SERVITO

JR - Comm, il potente programma telematico per Amiga, è disponibile gratis; o quasi...

Finalmente, dopo parecchi mesi di preparazione, testing e di pre-release ufficiali (si è arrivati fino alla versione 0.99r), finalmente è pronta la **1.0** dell'eccezionale pacchetto telematico **JR-Comm**, scritto integralmente da **John P. Radigan** autore dl uno dei più significativi eventi telematici dedicati all'Amiga.

Il pacchetto contiene, oltre al programma e ad alcuni **fonts** di cui necessita, un secondo programma, chiamato **JR-edit**, il cui principale scopo è quello di convertire le rubriche telefoniche della vecchia versione 0.94 nella nuova 1.0.

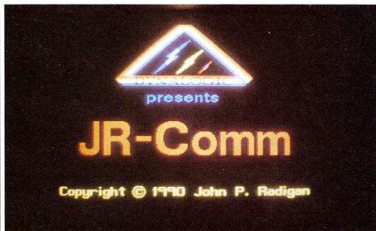
Questa è una raffinatezza di un certo rilievo, propria di programmi professionali di alto livello che sicuramente risparmierà ore di noiosa trascrizione ai possessori della precedente release.

JR non è particolarmente esigente: può utilizzare **qualunque configurazione** di Amiga e **qualunque tipo di modem**, purchè **Hayes** compatibile. Bisogna tuttavia tenere presente che chi possiede solamente il fatidico mezzo mega di memoria è soggetto alle solite restrizioni cui accenneremo.Caricato il programma, ci accoglie un'insolita schermatina colorata, che sfugge dopo qualche secondo, per introdurci nel modo terminale vero e proprio. L'ultima riga dello schermo è occupata dalla cosiddetta linea di stato che, oltre a mostrare i **settaggi** correnti del modem, ospita un **orologio** ed un **timer** azzerabile automaticamente ad ogni nuova connessione per determinare il tempo trascorso. Sulle prime righe superiori della borderless window (finestra senza bordi) a sfondo nero, troviamo il solito messaggino di Copyright, la data di rilascio e poco più. Tutto qua. Basta un tocco al pulsante destro del mouse per rivelare i menu a tendina e scoprire così la complessità e le molteplici funzionalità del programma. Nel riquadro sono riportate le voci di menu disponibili con una brevissima spiegazione / traduzione. Sorvoliamo velocemente il primo menu (**Project**) che contiene funzioni assai tipiche per gli utenti Amiga. Tra questi un comodissimo file **About** che spiega le condizioni di utilizzo del programma, i **Load** / **Write defaults** che servono per salvare, ed eventualmente ricaricare, impostazioni standard (che vengono caricate automaticamente dopo il lancio del programma) ed infine il classico **Quit** che non necessita di commenti.

Menu Phonebook

Attraverso questo menu si può accedere ad una completa **rubrica** telefonica. Selezionando la voce **Directory**, infatti, si apre un requester che possiede lo spazio per visualizzare quindici nominativi di bbs con relativi numeri telefonici; naturalmente se ne possono memorizzare molti di più, visualizzabili mediante un comodo gadget proporzionale. Molto interessante è la procedura per chiamare una o più banche dati: basta, nel caso interessi contattare una determinata bbs, un doppio click sul suo nominativo ed il gioco è fatto. Se, invece, desideriamo chiamare più banche dati, sarà sufficiente cliccare una volta sola sui nominativi nell'ordine che più ci aggrada. In caso di linea occupata verranno chiamate le rimanenti, una dopo l'altra, finchè qualcuna risulti libera.

In fase di composizione del numero, il grande requester si chiude lasciando il posto ad uno più piccolo, che contiene il nome ed il numero telefonico della banca dati attualmente chiamata ed il nume-

ro tentativi di collegamento. Vi sono possibilità di interazione: con la barra spaziatrice si può saltare la chiamata attuale passando alla successiva, con il tasto Del si elimina una bbs dall'elenco precedentemente selezionato ed infine con Esc si interrompe il processo di chiamata.

Ma procediamo con ordine...

Innanzitutto bisogna inserire dati nella lista pigiando il gadget **Add** del requester PoneBook il quale farà comparire un nuovo requester che consente di introdurre, oltre al nome della bbs, al relativo numero di telefono e alla password, anche le configurazioni della seriale, del terminale e del protocollo di trasferimento files mecessarie, nonchè particolari macro associate ai tasti funzione e palette.

Un'altra funzione inedita è costituita dalla possibilità di autogenerazione di password.

Mediante un preciso algoritmo seguito dal programma è possibile, fornendo una parola in input, lasciarla codificare dal computer in complesse sequenze di caratteri in modo da renderla più "sicura".

Un'alta feature del phonebook è quella di associare, ad ogni bbs, anche un parametro opzionale riguardante il costo in scatti SIP al minuto: se viene inserito un valore, stimato consultando il cosiddetto Avantielenco degli elenchi telefonici, nella linea di stato, in basso, durante le connessioni, vedremo anche quanto stiamo spendendo, pur se in via approssimativa.

Sempre dal requester phonebook si può eseguire il **Sort**, l'ordinamento degli entry (un **entry** o record è costituito da un gruppo di dati inserito in un sistema di archiviazione; praticamente, nel nostro caso, rappresenta il nome della bbs con numero di telefono, password e altri parametri), che viene eseguito non solo in ordine alfabetico: a scelta può essere effettuato anche per ordine crescente di numero telefonico o in un ordine selezionabile manualmente.

Le altre funzioni del phonebook sono, oltre ai canonici **Load** e **Save**, il **Delete** per depennare una bbs dall'elenco, l'**Edit** che consente di modificare qualunque parametro di un'entry, il **Dial** che, in alternativa al doppio click del mouse, fa partire la composizione del numero telefonico di uno o più servizi telematici, ed infine l'**Unselect** che annulla una sequenza di chiamata automatica impostata.

Continuando l'esame dei menu, sempre in Phonebook, troviamo **Redial**, che ha la funzione di riprendere l'esecuzione delle chiamate automatiche dopo un'interruzione (ad esempio la connessione ad una bbs, un'errata interpretazione del modem o più semplicemente la pressione del tasto Esc).

Molto più interessante è la funzione **Send Password** che permette di inviare automaticamente la nostra password, memorizzata insieme al nome ed al numero telefonico del sistema chiamato. Basta quindi selezionare quel comando quando una bbs chiederà di inserire la password senza dovere spulciare tra biglietti e bigliettini come ai vecchi tempi.

Il menu Buffer

Senza dilungarsi inutilmente, diremo che è possibile azzerare il buffer degli ultimi caratteri ricetrasmessi (buffer di dimensione selezionabile, vedi avanti), attivare il modo di lettura del buffer, aprire e chiudere un **Capture** su disco, cioè la memorizzazione su supporto megnetico dei caratteri digitati e, comunque, interscambiati, con la possibilità di salvare anche il buffer e / o di aggiungere il capture ad un altro testo preesistente.

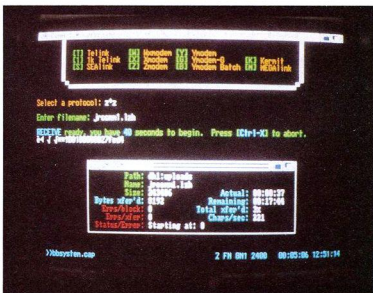
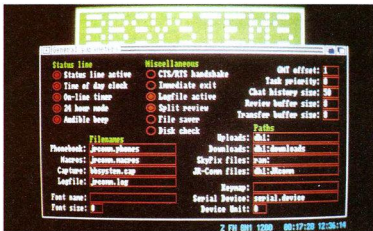
Il menu Transfer

Qui incontriamo il primo di vari requester di opzioni, richieste anche in fase in inserimento dei dati nel phonebook e si sceglie il protocollo di trasferimento da usare per default.

I protocolli disponibili sono 9: **CIS B+, WXmodem, Xmodem, Xmodem-CRC, Xmodem-1k, Ymodem, Ymodem-1k, Ymodem-g, Z-modem.**

Tra le venti opzioni selezionabili, di cui ben tredici dedicate allo **Z-modem**, parleremo solo delle due più importanti. Con **Auto Download**, il programma si accorgerà automaticamente quando il sistema remoto tenterà di inviare un file; con **Resume Transfer** si potrà rimediare all'interruzione del download di un file proseguendo il lavoro senza essere costretti a ricominciare da capo il trasferimento.

Trascurando **Upload** e **Download File**, che semplicemente attivano la ricetrasmissione di un file con il protocollo abilitato, spenderemo qualche parola in merito a **Ascii Send**.

Invece di editare *on line* (cioè *durante* un collegamento con una bbs) i messaggi da trasmettere, eventualmente, su qualche Bulletin Board System, risulta molto più conveniente, in termini di tempo (e, quindi, di scatti SIP) scrivere *prima* del collegamento *(off line)* i messaggi con un normale editor, e *successiva-*

mente scaricarli sulla bbs come se fossero dei comuni files.

Con Ascii Send verrà, appunto, richiesto il nome del file che verrà rapidamente spedito, via seriale, al sistema remoto facendogli credere che siate voi a scriverlo a velocità supersonica. Particolari parametri consentono di espandere le linee vuote (che verrebbero intese, da alcuni sistemi, come la fine del testo) in uno spazio, oppure stabilire pause tra carattere e carattere e/o tra linea e linea; tutto questo per venire incontro alle esigenze del sistema remoto cui vi collegate.

Il menu Options

Qui troviamo tutte le altre finestre di dialogo visualizzabili in fase di **Adding** nel phonebook. Iniziamo con **Serial** che, come facilmente intuibile, richiede i parametri della seriale: velocità da **300** a **57.600** baud, bit di dati e di stop, parità e duplex. E' poi la volta della voce **Modem** che serve per inserire gli argomenti necessari al "compositore telefonico intelligente", come lo chiama lo stesso autore, affinchè reagisca correttamente alle risposte del modem. Inoltre c'è lo spazio per la classica stringa di inizializzazione: una **stringa di comandi** Hayes, con la possibilità di inserimento di precise temporizzazioni, che vengono inviate al modem in fase di **Set-up** (inizializzazione) per fornire a quei modem, sprovvisti di memoria-tampone, dei settaggi di base che soddisfano le nostre (anche se elevate) esigenze.

Vanno inseriti anche i parametri del compositore telefonico intelligente: il **Redial Delay**, cioè il tempo in secondi che deve trascorrere tra due tentivi di collegamento andati a vuoto, ed il numero di **tentativi** da effettuare per trovare una linea libera.

Il menu Terminal

E' possibile selezionare il tipo di terminale, screen e altri parametri da impiegare nel corso delle nostre appassionate ore di telecomunicazione.

Come *emulazioni di terminali* sono disponibili: **TTY** (monocromatico, tipico dei sistemi Unix e Vax), **Amiga Ansi** (ideale per scambiare dati con altri Amiga), **VT100** (terminale ANSI abbastanza diffuso), **IBM mono** (8 tonalità di grigio per collegamenti a PC compatibili monocromatici), **IBM color** (16 colori di un PC compatibile, ideale per il 99% dei collegamenti, specialmente con sistemi Opus) ed infine l'incredibile **SkyPix**, un modo grafico, nato apposta per l'Amiga, che permette di trasmettere molta grafica e fonts (addirittura **immagini IFF**, ma attenti al tempo...) e consente di selezionare voci di menu via mouse. Quest'ultimo protocollo è da impiegare con sistemi che girano su sistemi **Amiga + Atredes Bbs**.

A quest'ampia possibilità di emulazioni, si associa tutta la gamma di screen Amiga: dai **2** ai **16** colori (da 1 a 4 bitplanes), dal **WorkBench** all'**Interlace**.

Avendo a disposizione solo 500 Kram (**Amiga 500 ridotto ai minimi termini...**), sarete costretti a usare non più di 8 colori (3 bit-planes), ma anche così vi troverete qualche volta senza memoria. Tranquilli, però: nessuna Guru Meditation!

Un semplice requester avverte dell'impossibilità di aprire una window richiesta o roba analoga.

Va notato che, selezionando particolari terminali, automaticamente vengono selezionati anche lo screen ed altri parametri che influiscono sul terminale stesso, o tipici dell'emulazione richiesta: è questo il caso dell'IBM color che implementa uno screen di 16 colori, il **blinking** (lampeggiamento) del cursore, il **wrapping** delle parole (mandata a capo automatica delle parole che superano il bordo destro), ecc.

Tra le principali opzioni selezionabili, oltre al citato blinking e wrapping, compare anche la cosidetta **IBM Doorway** e **Optimized Scroll**. Il primo abilita il doppio senso del tastierino numerico, trasmettendo, per esempio, al posto del 9, il **PgUp**, ecc. Risulta particolarmente comodo per i SysOp o chiunque debba utilizzare prodotti telematici che richiedono, ed offrono, l'impiego dei tasti "speciali".

Optimize scroll, invece, rimpiazza il rustico scroll di Amiga con una più efficiente e rapida routine di scrolling scritta dall'autore.

E non è ancora finita. Qualcuno ha già provato a fare un capture di una session (scusate il tecnicismo; per **Session** si intende una "seduta", un collegamento) in grafica ANSI? Si ottiene un file quasi senza senso, che va pulito dai caratteri grafici di **Esc**ape con **Ansi2asc** o programmini - filtro analoghi. Bene, con JR-Comm basta abilitare i capture filters: al resto pensa "lui".

Con la voce **Macros**, il cui requester è in grado di memorizzare ben *quaranta* macro diverse, si possono assegnare, ai dieci tasti funzione (eventualmente premuti insieme ad altri, cioè **Shift**ati, **Control**lati, **Alt**ati o... normali) particolari sequenze di caratteri.

Penultima opzione del menu è **Palette** che, come intuibile, modifica i colori dell'interfaccia utente a patto che non si sia scelta, da Terminal, la modalità WorkBench.

General è l'ultima opzione del menu; consente di immettere i parametri generali di JR. Alcuni di questi riguardano la **Status Line**: innanzitutto se la si vuole o meno. Inoltre si può inserire / disinserire l'**Orologio** (12 o 24 ore) ed il **Timer**. E' possibile scegliere tra un *beep* acustico o il classico *flash* dello screen. Sempre tramite comodi gadget booleani si può attivare il **Log** dei collegamenti o attivare lo **Split Review**. Questa possibilità è un'altra delle implementazioni del prodotto: mentre siamo in collegamento, selezionando l'opzione **Review Buffer** del menu Capture, lo schermo si divide a metà e, mentre nella parte inferiore continuiamo a rice-trasmettere dati, la parte superiore consente di visualizzare il contenuto del buffer.

Si possono quindi modificare le dimensioni del buffer e la capacità della **Chat History**.

Per terminare, si possono inserire i nomi che JR impiega per salvare la configurazione, il phonebook, il log, il capture, oltre ai path di up/down load ed al **Nome del Serial Device**, praticamente indispensabile per schede multiseriali, modem su scheda per Amiga 2000, e per modem veloci (più di 9600 baud).

Il menu Misc

Con questo possiamo resettare il timer (**Reset Timer**), inviare una pausa per favorire la lettura di un testo (**Send Break**), riappendere la linea telefonica, interrompendo una comunicazione (**Hangup Modem**), cancellare lo schermo (**Clear Screen**), stampare una schermata su carta (**Print Screen**) e, udite udite, stampare il modulo di registrazione per pagare l'utilizzo del programma (da inviare all'autore...).

Il menu Mode

L'ultimo menu offre una serie di opzioni booleane (attivare / disattivare). Queste vanno dalla possibilità di mettere e togliere la **stampante** on-line per avere una sorta di capture su carta, dalla chat-line, dall'**Hex - output** per analizzare *tutti* i caratteri ricevuti, anche quelli di controllo normalmente non visibili, al comodissimo **Doorway - Mode** per abilitare l'emulazione della tastiera dei compatibili PC (PgUp, PgDn, ecc.), alla tipica possibilità di mettere / togliere la titlebar, quella dello screen.

Tiriamo le somme

Dopo la velocissima corsa tra le opzioni del menu, è arrivato il momento delle considerazioni e delle impressioni

Project Progetti
About JR-Comm Informazioni su JR-Comm
Load defaults Carica dati standard
Write defaults Scrive dati standard
Quit JR-Comm Termina sessione di JR-Comm

Phonebook Rubrica Telefonica
DirectoryIndice del dischetto
Re-dial Ricompone numero
Send password Invia parola d'ordine

Buffer Buffer
Kill buffer Cancella la memoria di transito
View buffer Visiona il buffer
Open capture Salva il testo in ingresso
Open w/save Salva buffer e testo
Append capture Aggiunge in coda il testo in ingresso
Append w/save Appende buffer e testo
Close capture Interrompe salvataggio testo

Transfers Trasferimenti files
ParametersParametri
Upload file Trasmette file
Download file Riceve file
ASCII send Invia testo ASCII

Options Opzioni
SerialPorta seriale
ModemModem
TerminalTerminale
MacrosMacroistruzioni (F1, F2, ...)
PalettePalette di colori
GeneralDati generali

MiscVarie
Timer reset Resetta il timer
Send break Invia una pausa
Hangup modem Interrompe la linea
Clear screen Cancella lo schermo
Print screen Stampa lo schermo
RegistrationStampa il modulo di registrazione

ModesModi
PrinterStampante in linea
Chat line Particolare metodo di input
HEX output Output esadecimale
IBM doorway Tastiera Amiga = IBM/PC

I menu del programma JR - Comm e la rispettiva traduzione in italiano. Si noti la ricchezza dei comandi e le numerose modalità di impiego.

d'uso. Da quanto detto è comprensibile che la **Doc**umentazione è lunga **200K**. L'uso del programma è il più felice che si possa incontrare usando Amiga, telematicamente parlando. E' comodissimo da usare ed è tutto alla portata di mano. I lettori alle prime armi potrebbero scoraggiarsi leggendo la gran quantità di informazioni sulle funzioni possibili. Al contrario, **non è affatto complicato impiegare JR per comunicare via modem**: pur disponendo di funzioni e parametri sofisticati, altamente professionali, il programma riesce a mantenere quella semplicità d'uso necessaria per essere adoperato da chiunque, consentendo una padronanza graduale.

Non abbiamo però ancora parlato a fondo della caratteristica forse più importante e nobile di tutte: Jack Radigan ha inserito il suo programma nella categoria degli **Sheraware**. Ciò sta a significare che è liberamente copiabile, pur se è richiesto un modico contributo economico; questo, in ogni caso, andrà corrisposto all'autore esclusivamente se, dopo **un mese** di tempo di utilizzo per consentire la valutazione del prodotto, continuiamo ad impiegarlo ritenendolo valido.

Pur se può sembrarvi ingenuo, teniamo a sottolineare l'importanza di pagare le licenze d'uso di tutti i validi programmi shareware che usate abitualmente. Solo in questo modo si può evitare che diventino commerciali.

Ogni programmatore, seppur dilettante, sa quanto lavoro e fatica stia dietro la programmazione; è quindi giusto premiare persone oneste che condividono questa passione, senza bramosia di denaro.

La licenza d'uso, comprensiva di un dischetto con la versione più aggiornata del programma e delle spese di spedizione, è di soli **40 dollari** statunitensi. Se solo pensiamo che il più recente programma commerciale per modem (K-Comm della Kuma, che possiede solo il vecchio protocollo X-modem...) costa intorno ai 50 Dollari e che altri in Italia vengono venduti a prezzi esorbitanti (A-talk III L. 160.000) pur essendo ormai molto vecchi, ci sembra il caso di non dilungarci inutilmente elogiando un prodotto molto valido. Naturalmente, per ogni problema o quesito su JR, oltre a chiamare direttamente le bbs supporters ufficiali negli Stati Uniti(!), l'autore del presente articolo sarà ben disposto a rispondervi.

Contattatelo su **Bbsystem** (Tel: **02 / 57.60.52.11**) oppure, se preferite i canali **Matrix** e/o **Echo** della **FidoNet**, lo trovate anche su **Euro Elettronica BBS** (**2: 331 / 203 - 0373 / 86966**) e sulla nuova Soft House Club (2: 331 / 202 - 0373 / 273188).

di Domenico Pavone

MUSICA, MUSICA!

Un eccezionale pacchetto hardware e software che trasforma l'Amiga in una consolle professionale in grado di creare sofisticati brani musicali

Amiga uguale grafica, ma Amiga uguale anche suono e musica.

Questo ritornello, ormai sulla bocca di tutti, non finirà mai di stuzzicare l'orgoglio di chi si pregia di possedere il gioiello di casa Commodore. Magari alla faccia dell'amico non amigo, quello che per la stessa cifra di acquisto si è ritrovato tra le mani poco più di un giocattolo che ogni tanto emette un pietoso beep.

Al cospetto di tale meraviglia tecnologica, pare che qualche aborigeno di passaggio sia rimasto impietrito dalla sorpresa.

Poi, passato lo sbigottimento, sia corso ad acquistare un Amiga.

Ma non lasciamoci prendere la mano dalla fantasia, nè dalla megolamania, anche perchè la cosa presenta un rovescio della medaglia. O meglio, presentava.

Un normale utente di Amiga si trova infatti bombardato da centinaia di games, demo, o anche utility che sfoggiano colonne sonore di tutto rispetto, paragonabili a qualunque altra fonte sonora degna di tale nome (hi-fi, megatape, eccetera).

E proprio in questo, subdolamente, si nasconde una fonte di frustrazione non indifferente. Sì, tutto bello, ma come lo sarebbe di più se si potesse far riprodurre, ad Amiga, quella svisata degli U2 che ci piace tanto, o fargli ripetere all'infinito che "Nothing compares to you", lasciando poi all'immaginazione (o a qualche Pic ben ammaestrata) il primo piano della O'Connors.

Risultati, questi, non certo raggiungibili con i miseri comandi Sound e Wave del Basic, oppure stilando qualche decina di migliaia di singoli codici macchina, o anche solo affidandosi ai pur validissimi tool musicali esistenti sul mercato.

A meno di non essere superprogrammatori, supermusicisti, e... supermasochisti.

Già; anche ammettendo di essere in grado di farlo, perchè occupare così gli anni migliori della propria esistenza, quando si può ottenere lo stesso risultato in poco più di qualche secondo?

La parola d'ordine, insomma, è una: digitalizzare.

Cosa, questa, ormai da tempo tecnicamente di larga diffusione, non altrettanto alla portata di tutti sotto l'aspetto economico. Ma i tempi cambiano, i prezzi anche, ed ecco sul mercato un prodotto facile da usare, e dal prezzo più che abbordabile.

Stereo Professional Sample Studio

L.185000 iva compresa.
In vendita, anche per corrispondenza, presso
Flopperia
Viale Monte Nero, 31
20135 - Milano
tel. (02) 55.18.04.84

La scatola magica

Si sta parlando del package hardware/software **Stereo Professional Sample Studio,** prodotto dalla **Datel Electronics**, ben nota agli utenti di Amiga per la sua cartridge **Action Replay**.

Del prodotto esistono due versioni: una per i modelli 500 e 2000, ed una per i non pochi aficionados che continuano a strizzare il vecchio Amiga 1000.

La sezione hardware è rappresentata da un contenitore molto spartano, dalle dimensioni abbastanza ridotte, dal quale fuoriesce, nella parte posteriore, un cavo multipista collegato ad un connettore da inserire nella porta parallela di Amiga, quella comunemente adoperata per la stampante.

Sul lato opposto, fanno bella mostra di sè tre ingressi che consentono la connessione ad una fonte audio mediante uno spinotto Din a 5 poli, o tramite comuni jack stereo. Questi ultimi, in particolare, prevedono l'input da un microfono (e quindi la possibilità di registrare la propria voce), oppure da una qualunque altra fonte sonora: registratore, hi-fi, radio, Tv, e chi più ne ha più ne metta.

Ai fini di una riproduzione il più fedele e pulita possibile, va prestata una certa cura nel collegamento alla fonte esterna. Nulla vieta che ci si arrangi a suon di forbici e nastro isolante per adattare uno spinotto-jack ad un cavetto qualunque, ma è buona norma adoperare cavi schermati, ed evitare connessioni estemporanee che potrebbero provocare l'innescarsi di frequenze decisamente sgradevoli all'ascolto.

I cavi per il collegamento esterno non sono compresi nella confezione, cosa peraltro ovvia se si considera la diversità delle possibili fonti sonore.

Una volta effettuati i collegamenti **a computer spento**, non resta che dare tensione ad Amiga, e lanciare il dischetto a corredo del package, dotato di autoboot. Va detto, prima di ogni altra cosa, che il digitalizzatore è fornito di un manuale descrittivo di una trentina di pagine, in inglese. Cosa, quest'ultima, che non deve impensierire più di tanto i non anglofili: tutto il software è di estrema semplicità, manipolabile unicamente a suon di mouse, ed intuitivo tanto nelle manovre eseguibili direttamente "a video" quanto in quelle implementate da menu pulldown.

Dopo lo start, una schermata di presentazione (ovviamente dotata di colonna sonora) consente di selezionare l'accesso a due diversi applicativi: il **Sampler**, che si occupa della digitalizzazione vera e propria, ed il **Jammer**, un semplice sequencer a quattro voci per eventuali elaborazioni personali.

La parte del leone, come ovvio, la fa il Sampler, mentre il Jammer può essere considerato come un optional non indispensabile, ma non per questo da disprezzare.

Chi è già esperto nell'uso di software musicale, potrà eventualmente affidarsi anche a package più specializzati nel trattamento personalizzato del suono, grazie anche alla principale caratteristica del Sound Sample: quella di **salvare su periferica il risultato delle digitalizzazioni in file di formato IFF**, standard di facile manipolazione, e direttamente implementato, per esempio, dal **Deluxe Music Construction Set** della Electronic Arts.

Ma tralasciamo per ora il Jammer, e diamo un'occhiata al cuore dello stereo Sample Studio.

Il software

Dopo una clickata per selezionare il Sampler, si accede alla schermata di lavoro vera e propria, molto curata graficamente, e divisa orizzontalmente in due sezioni identiche che si occupano (quella superiore) del canale audio sinistro, e (quella inferiore) del destro.

Ognuna è caratterizzata dalla presenza di una **pulsantiera** dalle funzioni ben note a chi abbia adoperato almeno una volta un normale registratore a nastri. Sulla destra, invece, è presente un pannello di controllo comprendente uno **Zoom** e i comandi probabilmente più adoperati: **Play** e **Stop**, la regolazione del livello di **Input**, nonchè il **Volume** generale di ascolto.

Il primo istinto è subito quello di attivare l'unità esterna, che assumeremo essere uno stereo tape, e provare subito il digitalizzatore. Nulla di male, ma quasi certamente i risultati non sarebbero quelli sperati.

La fase sicuramente più delicata consiste infatti nel determinare il livello di input del suono. In parte per tentativi, ma coadiuvati da una delle più utili feature del Sample Studio: il **Realtime**.

Agendo come di consueto sul pulsante destro del mouse, è infatti possibile accedere, tra gli altri, ad un menu **Extras** comprendente un subitem **Realtime**.

Una volta selezionato, appare sullo schermo una griglia che, in pratica, svolge le stesse funzioni del visore di un oscilloscopio. Dato il via al tape, alla chitarra elettrica, o a quello che si è deciso assumere come fonte sonora, la

finestra di realtime mostrerà il segnale audio in tutte le sue frequenze.

Operativamente, bisognerà fare in modo che i picchi siano percepibili (una linea piatta indicherà l'assenza di un segnale di input), e che non vengano "tagliati" al raggiungere la metà della griglia.

Ultimato il test preliminare, si può poi passare alla registrazione vera e propria, ovvero alla immissione, nella Ram di Amiga, del brano prescelto.

Operazione che è possibile effettuare in due modi: clickando sul tasto di registrazione di un singolo canale (una mappa di tuti i tasti è chiaramente illustrata nel manuale), e quindi immettendo solo quel canale, oppure sfruttando il riquadro **Stereo Sample**. In quest'ultimo caso, verranno prelevati in toto entrambi i canali stereo.

Praticamente, tutto fatto. Lo schermo sparirà per qualche secondo, ripresentando poi la schermata precedente, arricchita della rappresentazione grafica del brano immagazzinato. Nella finestra di zoom, si potrà anche notare un dettaglio delle frequenze puntate da una specie di cursore che taglia verticalmente la "feritoia" principale nella quale è mostrato l'andamento generale del segmento sonoro.

Il cursore è spostabile mediante il mouse, consentendo la visualizzazione di ogni dettaglio nella finestra di zoom, il cui range può essere regolato a proprio piacimento grazie a una serie di gadget che consentono in pratica ogni tipo di manovra: movimento fine, amplificazione dello zoom, eccetera.

Ma questi, in fondo, sono dettagli che possono riguardare i professionisti del suono. A noi, miseri (si fa per dire) amigofili dagli scopi ben più abietti, interessa ascoltare qualcosa che possa essere definita musica dagli altoparlanti del monitor di Amiga.

Musica, Amiga!

Detto fatto. Se tutto è andato per il verso giusto, basta una clickata su grosso riquadro **Play**, e finalmente si potrà sentire il nostro bel computerone deliziare le più intime e sensibili cavità del nostro apparato auricolare.

Il brano verrà eseguito ciclicamente, per una durata variabile che può anche arrivare ad una trentina di secondi se registrato a velocità minima, cosa che però comporta un certo decadimento della qualità.

La velocità di esecuzione, tra l'altro, può essere modificata agendo direttamente su altri due pulsanti (disegnati), e indipendentemente sui due canali, creando in quest'ultimo caso l'illusione di uno sfasamento che può (forse) risultare interessante.

Non si pensi, comunque, che la durata del brano sia troppo limitata: a meno di non possedere una configurazione hardware superdotata (memoria a quintali, schede acceleratrici, eccetera), le risorse di Amiga vengono già impegnate notevolmente da una digitalizzazione di tale portata. La registrazione su file IFF di un solo canale, con estensione di una sola ottava (il minimo), occuperà tra l'altro circa 98 Kbyte, da moltiplicare per sette se si opta per le 5 ottave!

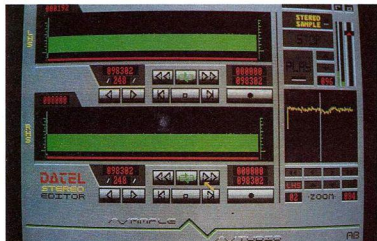
Insomma, per dirla terra terra, una durata simile basta e avanza.

Tanto più che, coordinando attentamente la fine del brano digitalizzato, si può creare l'illusione di una continuità notevole.

Tornando al nostro Sampler, c'è ancora da aggiungere che, anche in fase di ascolto, possono essere abilitati o disabilitati i singoli canali, o riportare all'inizio il brano (sempre singolarmente), e varie altre cosette da scoprire con l'uso pratico.

Dalla barra comandi di Intuition, si accede poi a 5 menu, che consentono di manipolare notevolmente il brano provvisoriamente memorizzato.

Le opzioni più ovvie (menu **Project**) riguardano il caricamento / salvataggio dei singoli canali audio. Come si è già detto, il Sampler può salvare i file in formato IFF, ma anche sotto forma di dati sequenzialmente disposti (**Raw**), con una testata (**header**) che specifica la dimensione del file e la frequenza adottata prima del save. In tal modo è, per esempio, possibile accodare i dati del brano ad una routine che li gestisca, naturalmente per i più esperti in programmazione.

Da un altro menu, **Sample**, si può poi decidere se i file salvati in formato IFF possono essere ricaricati e mandati in esecuzione una sola volta (**OneShot**), oppure riprendere l'esecuzione in un loop infinito (**Cont**). Da questo menu viene inoltre stabilito il numero di ottave da salvare nello stesso file, da trattare con attenzione, a meno che non si disponga di un hard disk.

Un file, che normalmente occuperebbe 100 Kappa di un floppy, se salvato dopo aver selezionato 5 ottave, necessiterebbe di cica 700 Kbyte, e si sta parlando di un solo canale!

Non poteva mancare, ovviamente, un menu (**Edit**) che consentisse di cancellare un canale, o di copiarlo nell'altro, nonchè di operare una miscelazione tra i due, con un effetto molto interessante.

E non è finita.

E' possibile anche (menu **Effects**) invertire un brano, nel senso proprio che l'ascolto risulterà uguale a quello che si otterrebbe facendo girare al contario il piatto di un giradischi (ancora e sempre agendo su un singolo canale stereo o su entrambi), ed il **Phaseshift**. Particolarmente interessante quest'ultima opzione, che permette di ottenere un effetto stereo anche da un segnale monofonico, cioè identico nei due canali, vuoi perchè registrato da una fonte non stereo, vuoi perchè frutto della copia di un canale nell'altro.

Il Sampler, in pratica, "sposta" di 128 byte uno dei due canali rispetto all'altro simulando un effetto eco che rende perfettamente l'idea dell' "ambiente".

Dell'ultimo menu, **Extras**, si è già appurata l'utilità del Realtime, ma sono presenti altre utili implementazioni, come il **Move** del contenuto di un canale nell'altro (o uno **Swap**, ovvero inversione), un test del suono, ed anche una **rappresentazione tridimensionale** dello sviluppo dell'onda sonora.

Per i più esperti, è anche disponibile da questo menu la possibilità di modificare molto intuitivamente, tramite il mouse, la forma d'onda di singole porzioni del brano, creando nuovi picchi positivi o negativi, o anche annullandone di già esistenti.

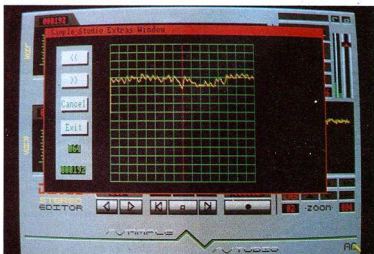
Inutile aggiungere che la tecnica della digitalizzazione, oltre che riprodurre su Amiga un certo brano, può essere impiegata per campionare suoni da elaborare in forma musicale tramite adeguati sequencer.

A tale scopo, come già accennato, può essere adoperato il programma **Jammer**, al quale si può accedere, oltre che dall'iniziale schermata di presentazione del Sample Studio, anche dal consueto workbench, disponibile dopo l'eventuale uscita dal Sampler.

Questa sezione del software consente di adoperare i suoni ricavati con il digitalizzatore per adoperarli come singole voci, da comporre in unità melodiche tramite una tastiera che occupa la metà inferiore del nuovo schermo di lavoro. Le note vanno immesse clickandovi sopra col mouse, con la possibilità di generare, oltre che singole note, anche accordi più complessi.

Non manca un **mixer** con cursori software, la ovvia ed indispensabile capacità di registrare la sequenza melodica che si immette, trasferibile poi su disco con il consueto item **Save** da menu, ed è anche implementato un ritmo di percussioni a frequenza variabile (la cosiddetta batteria elettronica, per intenderci).

D'altra parte il sequencer Jammer, assolutamente dignitoso ma non certo evoluto come altri specifici pacchetti esistenti sul mercato, è in pratica un "omaggio" della Datel agli acquirenti dello Stereo Sample Studio.

Un digitalizzatore, in definitiva, dal rapporto prezzo / qualità davvero eccellente, ma soprattutto in grado di mettere a disposizione dell'utente comune le non indifferenti risorse musicali di Amiga.

```
/* Cerchio animato Turbo-C by Mariani G.
*/

#include <stdio.h>
#include <conio.h>
#include <stdlib.h>
#include <graphics.h>

/* Costanti globali */
const char Up=72, Down=80, Left=75,
Right=77;
const char Esc=27, Cr=13;

/* Variabili globali */

/* Tasti premuti */
char ch[2];
/* Variabili per il cerchio */
int X,Y,R,OldX,OldY;
/* Array che contiene i movimenti */
char Movim[10000];
/* Contatore di movimenti */
int conta;

/* Prototipi di funzione */
void AutoMove (void);
void CaricaMovim (void);
void Grafica (void);
void ManMove (void);
void Move (char movimento);
void SalvaMovim(void);
void Tasto (void);

/* Mette lo schermo in grafica */
void Grafica()
{
int g_driver,g_mode;

/* Determina la scheda grafica presente */
detectgraph(&g_driver,&g_mode);
/* "c:\tc"= dir in cui sono i files .BGI */
initgraph(&g_driver, &g_mode, "c:\\tc");
cleardevice();
}

/* Movim. cerchio con array "Movim[]" */
void AutoMove()
{
int k;

if (conta==0) return;
Grafica();
```

Il listato scritto in Turbo - C Borland
Le linee del listato riportate in corsivo, che sembrano occupare due righe, vanno invece digitate su un unico rigo

presa in considerazione una controbarra, bisogna metterne due di seguito.

Lo schermo, messo in grafica, viene pulito con **cleardevice.**

Tasto

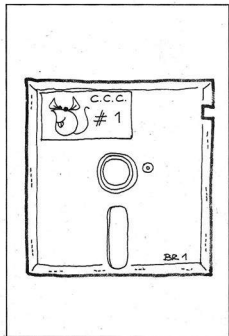
E' una procedura che aspetta la pressione di un tasto e ne mette il codice nel primo elemento dell'array ch (**ch[0]**). Se il tasto premuto appartiene ad uno dei tasti con codice **doppio** (tasti funzione, cursori, ecc.) il secondo codice viene trascritto in **ch[1]**.

ManMove

Accetta i parametri del cerchio, consente di muoverlo manualmente, memorizza i movimenti nell'array **Movim** e quindi ripropone in automatico la sequenza dei movimenti.

L'istruzione **clrscr** cancella lo schermo (in modo testo), e le due successive (**if kbhit**...) svuotano il buffer di tastiera per evitare di considerare tasti premuti accidentalmente.

In seguito viene proposta una mini-maschera che chiede i parametri del cerchio (**printf**) tramite **scanf**. La stringa di formato (**%d, %d, %d**) indica che i tre parametri sono interi e che devono essere separati da una virgola. Altri formati sono **%c** per i caratteri, **%s** per le stringhe, **%f** per i numeri in virgola mobile.

Le variabili **X**, **Y** e **R** sono passate sempre come indirizzo (notare il carattere **&** posto davanti al nome) perchè scanf deve modificarne il valore.

La successiva istruzione (**Grafica**) richiama la procedura vista prima.

Poi vengono salvate le coordinate iniziali nelle variabili **OldX** e **OldY** e quindi il puntatore dell'array **Movim** (**conta**) viene azzerato.

L'istruzione **setcolor** sceglie il colore con il quale disegnare (il colore **1** corrisponde al **bianco**), e **circle** disegna effettivamente il cerchio, alle coordinate X e Y, con raggio R.

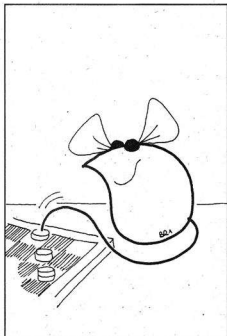
Il ciclo **do... while** contiene la chiamata alla procedura **Tasto**; se il tasto è una delle 4 frecce (**if ch[1]=left**...) viene richiamata la procedura **Move**, che effettua il movimento del cerchio; subito dopo il movimento (o meglio il tasto premuto) viene memorizzato nell'array Movim ed il puntatore conta incrementato di 1.

Il ciclo ha termine quando si preme il tasto Enter, quindi l'istruzione **closegraph** riporta lo schermo in modo testo.

AutoMove

Effettua l'animazione del cerchio, ossia il movimento automatico, secondo il contenuto dell'array Movim. Dapprima viene controllato l'array: se è vuoto (**conta==0**) la procedura ha subito termine.

Quindi viene messo lo schermo in Grafica e cancellato e le coordinate X e Y vengono ripristinate con i valori iniziali contenuti in OldX e OldY.

Tramite il ciclo **for**, da 0 al numero di movimenti effettuati, viene richiamata la routine **Move**, che prende come parametro il contenuto dell'array Movim ed anima il cerchio sullo schermo.

```
cleardevice();
X=OldX; Y=OldY;
for (k=0; k<conta; k++) Move(Movim[k]);
Tasto();
closegraph();
}

/* Carica i movimenti da disco */
void CaricaMovim()
{
FILE *stream;
char NomeFile[15],Temp[11];

if (kbhit()>0) getch();
if (kbhit()>0) getch();
conta=0;
clrscr();
printf("\nNome del file : ");
scanf("%s",NomeFile);
stream=fopen(NomeFile,"rb");
if (stream!=NULL)
{
fgets(Temp,10,stream); X=atoi(Temp);
fgets(Temp,10,stream); Y=atoi(Temp);
fgets(Temp,10,stream); R=atoi(Temp);
while (fgets(Temp,10,stream)!=NULL)
{ Movim[conta]=Temp[0]; conta++; }
OldX=X; OldY=Y;
}
fclose(stream);
}

/* Movimento del cerchio con i tasti */
void ManMove()
{
clrscr();
if (kbhit()>0) getch();
if (kbhit()>0) getch();
printf("\nInserisci X,Y,R del cerchio: ");
scanf ("%d,%d,%d",&X,&Y,&R);
Grafica();
conta=0;
OldX=X; OldY=Y;
setcolor(1);
circle(X,Y,R);

do
{
Tasto();
 if (ch[1]==Left || ch[1]==Right || ch[1]==Up ||
ch[1]==Down)
```

Il seguito del listato

I comandi riportati in corsivo vanno trascritti su un unico rigo e non su due, come qui riportato per esigenze di impaginazione.

```
{
Move(ch[1]);
Movim[conta]=ch[1];
conta++;
}
}
while (ch[0]!=Cr);
closegraph();
}

/* Muove il cerchio sullo schermo */
void Move(char movimento)
{

/* Cancella il cerchio */
setcolor(0);
circle(X,Y,R);

if (movimento==Left) X--;
if (movimento==Right) X++;
if (movimento==Up) Y--;
if (movimento==Down) Y++;

/* Ridisegna il cerchio */
setcolor(1);
circle(X,Y,R);
}

/* Salva i movimenti su disco */
void SalvaMovim()
{
FILE *stream;
char NomeFile[15],Temp[10];
int k;

if (kbhit()>0) getch();
if (kbhit()>0) getch();
if (conta==0) return;
clrscr();
printf("\nNome del file : ");
scanf("%s",NomeFile);
stream=fopen(NomeFile,"wb");
if (stream!=NULL)
{
itoa (OldX, Temp, 10); fprintf (stream,
"%s\r\n", Temp);
itoa (OldY, Temp, 10); fprintf (stream,
"%s\r\n", Temp);
itoa (R, Temp, 10); fprintf (stream, "%s\r\n",
Temp);
```

Il seguito del listato

I comandi riportati in corsivo vanno trascritti su un unico rigo e non su due, come qui riportato per esigenze di impaginazione.

Infine si aspetta la pressione di un tasto prima che closegraph riporti lo schermo in modo testo.

Move

Muove effettivamente il cerchio sullo schermo grafico. Serve sia per il movimento manuale che automatico.

Il parametro **movimento** contiene il valore del tasto premuto, oppure il valore contenuto in un elemento dell'array Movim, che memorizza la sequeza dei tasti premuti.

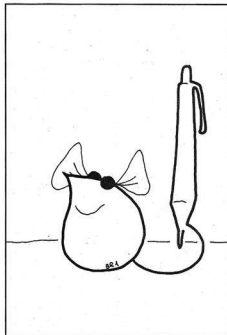
Dapprima il cerchio viene cancellato tramite **setcolor (0)** che imposta il colore **nero**; dal momento che anche lo sfondo è nero, qualunque cosa verrà tracciata sarà invisibile. La successiva istruzione circle disegna, appunto, un cerchio nero su sfondo nero, e quindi lo cancella.

Le quattro righe **if**, successive, controllano il valore del tasto premuto ed incrementano (o decrementano) opportunamente le coordinate X e Y. Come si nota, non è presente alcun controllo sui limiti dello schermo.

Le righe setcolor(1) e circle... ridisegnano il cerchio nella nuova posizione, questa volta con colore **1 (bianco)** e quindi visibile.

SalvaMovim

Registra i parametri del cerchio ed i movimenti effettuati su disco.

I due **if kbhit**... servono, come visto prima, per svuotare il buffer di tastiera. L'istruzione **if conta**... permette di terminare immediatamente la procedura nel caso in cui l'array Movim non contenga alcun movimento.

Clrscr cancella lo schermo prima che venga richiesto il nome del file (scanf ...). Inserito il nome, il file viene aperto tramile **fopen**, in modo write (**w**) e binario (**b**). Nella variabile **stream** viene trascritto il puntatore al file. Nel caso sussista qualche problema nell'apertura del file (nome scorretto, ecc.) questo puntatore viene messo a **NULL** e controllato dal ciclo successivo **if stream...**, che consente la scrittura del file solo in caso di apertura andata a buon fine.

Le istruzioni **itoa** convertono il valore intero passato (OldX, OldY, R) in una stringa (**Temp**). **Fprintf** è l'equivalente di **printf**, ma opera su files invece che sul video, e in questo caso registra i valori interi convertiti. E' da notare che i primi tre sono OldX, OldY e R; subito dopo vengono scaricati i parametri del cerchio.

Il successivo ciclo **for** permette di salvare tutti i movimenti contenuti nell'array **Movim**. E' da notare che tutte le stringhe di formato contenute in fprintf contengono, alla fine, la sequenza di caratteri **\r\n** che aggiunge, in coda alla riga salvata, il carattere **Ascii 13** (ritorno carrello) ed il carattere di nuova linea (**Ascii 10**); ne consegue che ogni parametro è salvato su di una singola riga del file. Ce ne possiamo accorgere impartendo, da ambiente Dos, il noto comando **Type Nomefile**.

L'istruzione **fclose** chiude il file, specificato nel nostro caso da **stream**.

CaricaMovim

Carica i parametri del cerchio ed i suoi movimenti da un file precedentemente salvato con **SalvaMovim**. Allo stesso modo di SalvaMovim, il buffer di tastiera viene svuotato, viene richiesto il nome del file da caricare e quindi viene azzerato il puntatore **conta** e aperto il file, questa volta in lettura (**r**).

Se il file esiste (puntatore stream diverso da NULL) vengono caricati, tramite **fgets**, le prime tre righe del file, quindi, tramite **atoi**, vengono convertite in numeri interi e trascritti nelle variabili X, Y, R, che definiscono i parametri del cerchio.

I movimenti vengono caricati tramite un ciclo **while**, che continua fino a quando l'istruzione fgets fornisce NULL come risultato (fine del file). I movimenti sono caricati uno alla volta, quindi messi nell'array Movim, ed il puntatore **conta** viene incrementato ogni volta di 1, grazie alla comoda forma sintattica, tipica del C, *"nome variabile più più"* (**conta++**).

Alla fine del caricamento le variabili OldX e OldY vengono inizializzate con i valori di X e Y, ed il file viene chiuso (**fclose stream**).

Main Program

E' questo il programma... principale. Tramite una successione di **printf** visualizza il menu iniziale, ed il richiamo alla routine **Tasto** permette di effettuare la scelta.

A seconda di questa verranno opportunamente richiamate le routines viste prima, ed il ciclo **do..while** del main program continuerà fino alla pressione del tasto Esc.

```
   for (k = 0; k< conta; k++) fprintf (stream,
"%c\r\n", Movim[k]);
   }
   fclose (stream);
   }

   /* Attende la pressione di un tasto */
   void Tasto ()
   {
   ch[1]=0;
   ch[0] = getch();
   if (ch[0]==0) ch[1]=getch();
   }

   /* Main program */
   main()
   {
   conta=0;
   do
   {
   clrscr();
   printf ("CERCHIO ANIMATO\n\n");
   printf ("1 - Movimento con tasti\n");
   printf ("2 - Movimento automatico\n");
   printf ("3 - Salvataggio movimenti\n");
   printf ("4 - Caricamento movimenti\n");
   printf ("ESC:Fine programma\n\n");
   printf ("Scegli : ");
   Tasto();
   if (ch[0] == '1') { ManMove(); AutoMove();
ch[0]=0; }
   if (ch[0]=='2') { AutoMove(); ch[0]=0; }
   if (ch[0]=='3') { SalvaMovim(); ch[0]=0; }
   if (ch[0]=='4') { CaricaMovim(); ch[0]=0; }
   }
   while (ch[0]!=Esc);
   }
```

Parte finale del listato

I comandi riportati in corsivo vanno trascritti su un unico rigo e non su due, come qui riportato per esigenze di impaginazione.

di Lorenzo Emilitri

AMIGA IN MUSICA

Come usare uno dei migliori pacchetti applicativi per il 16 bit della Commodore; e perchè

Una delle ragioni che dovrebbero spingere il potenziale acquirente di un nuovo computer a scegliere **Amiga** è la sua capacità di (ri)produrre suoni e musica in stereo con elevato livello di fedeltà. Spesso, però, dopo aver portato a casa il voluminoso imballo ed aver effettuato tutte le connessioni necessarie, ci si accorge che i programmi musicali che ci sono stati più o meno regalati al momento dell'acquisto, pur producendo risultati più che dignitosi, rispetto agli altri computers, non riescono a reggere il confronto con le musiche contenute, ad esempio, nei videogiochi.

Ciò accade in quanto molti programmi musicali sono pensati per "uscire" su una **tastiera MIDI**; utilizzano il pentagramma e permettono accordi di più di quattro note per volta, ma poi non riescono ad eseguirli; non permettono quasi mai di riascoltare le musiche "fuori" dal programma che le ha generate; hanno a disposizione strumenti ben poco realistici e spesso difficili da programmare (selezione **ADSR**); non dispongono della possibilità di inserire effetti di una certa complessità (voci, fischi, campanelli, accordi, esplosioni, ecc.); non sfruttano adeguatamente l'effetto stereofonico.

Per sfruttare l'Amiga fino in fondo dobbiamo utilizzare programmi progettati su misura per questo computer.

Quelli che, secondo noi, sono i migliori, anche (e soprattutto) per i principianti, sono quattro: **SoundTracker, Oktalyzer** e **Future-Composer**. Occupiamoci del primo.

Suoni e musiche in otto bit

Per comprendere sia il funzionamento dei programmi, sia i motivi che hanno indotto gli autori a scriverli nella maniera in cui ce li ritroviamo, è necessaria una breve premessa sul funzionamento dell'hardware di Amiga, almeno per quanto riguarda la gestione sonora.

All'interno, infatti, vi sono **quattro canali**, corrispondenti ad altrettanti **convertitori D/A** a 8 bit, (il canale **0** e il **3** sono collegati con l'uscita di **sinistra**, il **2** e l'**1** con quella di **destra**) che si occupano di riprodurre i suoni presenti in memoria. Un aspetto importante è che, a differenza di molti altri computer, produrre suoni con Amiga significa sempre e solo riprodurre a velocità differenti suoni presenti in memoria sotto forma di tabelle di dati ottenute, in genere, tramite un campionatore.

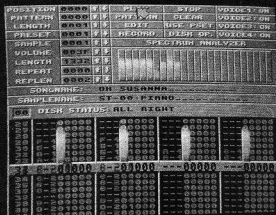
Ciò significa che per Amiga è la stessa cosa riprodurre un Fa Diesis di uno strumento scelto oppure farvi riascoltare, a velocità differenti, il ritornello della can-

Nota	3	2	1
Do	$0d6	$1ac	$358
Do#	$0ca	$194	$328
Re	$0be	$17d	$2fa
Re#	$0b4	$168	$2d0
Mi	$0aa	$153	$2a6
Fa	$0a0	$140	$280
Fa#	$097	$12e	$25c
Sol	$08f	$11d	$23a
Sol#	$087	$10d	$21a
La	$7f	$0fe	$1fc
La#	$078	$0f0	$1e0
Si	$071	$0e2	$1c5

Tabella 1: Periodi di campionamento di SoundTracker

Suoniamo con SoundTracker

Per darvi una dimostrazione pratica di come funziona Soundtracker, proviamo ad inserire il ritornello della famosa canzone "Oh Susanna" le cui note si trovano nel riquadro apposito.

Per prima cosa, carichiamo Soundtracker e settiamo il volume al massimo. Quindi scegliamo lo strumento da utilizzare. Supponendo di usare un pianoforte, posizionamo la freccia del mouse sul rigo **Preset** e, spostandola in corrispondenza delle due frecce, premiamo il tasto su quella verso l'alto finchè, arrivati alla lettera **P**, avanziamo lentamente per far apparire la parola "**Piano**". Ne esisteranno probabilmente parecchi, ma a noi ne serve uno di lunghezza accettabile (4000 - 6000 bytes). Per caricarlo in memoria basta premere **Use Pset**: il computer chiederà di introdurre il disco giusto per caricarlo.

Se non disponete dei dischi contenenti il Preset, niente paura: caricate un brano qualsiasi (almeno questo dovete averlo!) ed in seguito, fingendo di apportare modifiche, ne cancellate le note dopo aver selezionato il gadget **Edit**.

Premendo i tasti cui corrispondono le varie note (seguendo la tabella specifica riportata in queste pagine, ed assicurandosi di aver selezionato **Edit**), dovrebbe essere possibile ascoltarle direttamente, senza esser costretti a "costruirle" una per una settando i vari parametri. Se non ci soddisfa, è sufficiente selezionare un altro strumento tramite Preset e ripetere da capo.

Trovato lo strumento adatto, premiamo il gadget **Edit** con il mouse (dovreste aver capito che questo gadget è indispensabile per iniziare a comporre...); la freccia del topo cambierà colore ed apparirà il cursore nel pattern alla posizione **0**. Premiamo ora **F1** per indicare che vogliamo utilizzare le prime due ottave (con **F2** si selezionano le altre due disponibili) ed inseriamo le note del riquadro di "Oh Susanna" lasciando, sotto di esse, il numero di spazi indicato nel riquadro stesso; l'editing è molto semplice: è sufficiente utilizzare i tasti indicati nella tabella 2 che corrispondono alle varie note musicali. Ad esempio, premere **J** per la nota **A#** (notazione italiana: **La#**) ed i tasti cursore per spostarsi, mentre **Del** serve a correggere gli errori.

Per sapere se il lavoro procede bene, premiamo **Pattern** per ascoltare ciò che abbiamo inserito e quindi di nuovo Edit per correggere o ampliare.

Alla fine, se quanto realizzato vi piace, è sufficiente premere **Disk Op** e quindi **Save Song** per salvare il brano, non dimenticando di inserire il nome della musica!

zone preferita. Memoria permettendo, sarebbe teoricamente possibile inserire in Amiga un'intera musica digitalizzata e riprodurla con una fedeltà molto prossima a quella del vostro stereo. Uno degli aspetti da tenere in considerazione è che i suoni campionati hanno il brutto vizio di essere tremendamente lunghi, ed è molto facile farsi prendere la mano e ritrovarsi con un brano lungo diverse centinaia di Kilobyte.

Il secondo aspetto da tenere in considerazione è che, come accennato, le differenti altezze sonore sono determinate unicamente dalla velocità a cui viene riprodotto il campione, e che salire di un'ottava, a causa di una ben nota legge fisica, significa raddoppiare la velocità di esecuzione e quindi dimezzare il tempo in cui il campione viene suonato (vedi tabella 1).

Ad esempio, un suono che, per essere eseguito alla velocità Do1, impiega 10 secondi, ne impiegherà 5 alla velocità Do2 e solo 2.5 alla velocità Do3!

Per questo, volendo riprodurre strumenti con una notevole estensione (come il pianoforte) è necessario disporre di più campioni "presi" ogni una o due ottave e suonare il campione giusto al momento giusto; di fatto, disponendo di due campioni **Do1** e **Do3** di un pianoforte digitalizzati alla velocità Do1, per ottenere un **Do4** accettabile è necessario suo-

Angl.	Ital.	1	2
C	Do	Z	Q
C#	Do#	S	2
D	Re	X	W
D#	Re#	D	3
E	Mi	C	E
F	Fa	V	R
F#	Fa#	G	5
G	Sol	B	T
G#	Sol#	H	6
A	La	N	Y
A#	La#	J	7
B	Si	M	U

Tabella 2. La prima colonna rappresenta la notazione anglosassone, la seconda quella italiana; la terza e la quarta, rispettivamente, la prima e la seconda ottava. Premendo il tasto **F1** *si editano le prime due ottave; con* **F2** *la seconda e la terza. (n.b.: il simbolo* **#** *corrisponde all'innalzamento di un semitono;,* **"b"** *all'abbassamento. Ciò significa che i simboli* **Sib** *e* **La#** *sono equivalenti.*

nare lo stesso campione Do3 alla velocità Do2.

Ogni voce dispone, inoltre, di un registro **volume** in cui si possono specificare valori compresi tra **0** e **64**; il massimo volume di un campione pari a +/- 127 produce in uscita una tensione di +/- 0.4 volt, mentre specificare il volume 1, causa un'attenuazione di -36.1 decibel.

E' inoltre possibile, sempre agendo via hardware, chiedere ad una voce di modulare il periodo e/o il volume di un'altra; ciò permette di ottenere effetti particolari quale, ad esempio, l'effetto tremolo.

Soundtracker

Il SoundTracker (d'ora in poi **ST**), o meglio **The SoundTracker System** come lo chiamano gli appassionati, è stato uno dei primi programmi musicali a venire alla luce e affonda le sue origini nella preistoria di Amiga. E' stato scritto per la prima volta dal noto musicista elettronico **Karsten Obarski**, ed ha immediatamente riscosso un notevole successo sia tra gli hackers che gli appassionati in generale.

Sp	1	2	3	4
3	A#1	A#1	D-1	F-1
3	A#1			
1	D-2			
3	D-2			
1	D-2			
1	C-2	F-1	A-1	C-1
1	C-2			
1	A-1			
1	F-1			
5	G-1	C-1	F-1	A#1
0	F-1			
0	G-1			
1	A-1	F-1	A-1	C-1
1	C-2			
2	C-2			
0	D-2			
1	C-2			
1	A-1			
2	F-1			
0	G-1			
1	A-1			
1	A-1			
1	G-1	C-1	E-1	A#1
1	G-1			

Note di "Oh, Susanna!"
Ecco le note da inserire; gli spazi, in effetti, vanno inseriti dopo la nota cui si riferiscono (vedi articolo per maggiori dettagli). La prima colonna si riferisce agli spazi; le altre quattro (numerate con 1, 2, 3, 4) si riferiscono alle corrispondenti voci.

Ne sono state scritte almeno una decina di versioni, forse non completamente compatibili le une con le altre, soprattutto a livello di comandi. Qui ci riferiamo al SoundTracker **v 2.3** di MnemoTron, che permette di tenere in memoria, contemporaneamente, ben 32 suoni lunghi sino a 32K ciascuno.

Perchè funzioni, sono necessari anche altri dischi, chiamati **ST-xx**, dove **xx** è un numero di due cifre, tra **01** e **99** (su cui si trovano gli strumenti utilizzati) ed un programma del tipo **PresetEditor** (ne esistono di comodissimi) che, partendo dalle directory dei dischi ST, costruisca il file PresetList (**PLST**), necessario al SoundTracker.

❑❒

"Dentro" SoundTracker

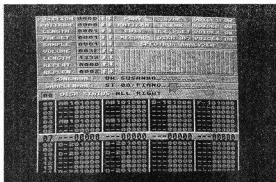
SoundTracker si presenta con lo schermo diviso in due parti: quella superiore è occupata da **gadget** (= rettangolini da clickare) per selezionare strumenti e patterns e gestire l'interazione con l'operatore; quella inferiore presenta il pattern vero e proprio, o almeno una sua parte; al centro si trovano il nome della **song** corrente, il nome del **sample** prescelto ed il numero del pattern che si sta editando.

Alla base di SoundTracker vi sono i **Patterns**, tabelle)delle dimensioni di 64 righe per quattro colonne ciascuna) che rappresentano una parte del brano musicale.

Successivamente, tramite le opzioni **Position, Pattern** e **Length** si impostano il numero di pattern totali che devono essere suonati (Length), e il numero di pattern (Pattern) corrispondente alla posizione selezionata.

Ad esempio, se la nostra musica è formata da sette "pezzetti" consecutivi e, in particolare, dal pattern n. **0** ripetuto **due** volte, poi dal n. **2** (una sola volta), dal **3** ripetuto **due** volte, di nuovo dallo **0** e infine dal n. **4**, dapprima imposteremo Length a 7, poi setteremo Position a 0 ed inseriremo 0 in Pattern, quindi incrementeremo Position e metteremo un altro 0 in Pattern; incrementeremo di nuovo Position inserendo 2 in Pattern e così via fino ad avere completato lo schema della nostra musica.

I pattern vengono suonati in orizzontale, ovvero prima la riga numero 0, poi la numero 1 e così via sino alla 63. Le note presenti in una riga vengono suonate contemporaneamente. Per determinare la lunghezza di un suono, si lasciano tanti spazi, sotto di esso, quanto si vuole che duri.

Se, ad esempio, una nota da 1/4 occuperà quattro caselle in verticale (una piena, con la nota stessa, più tre vuote), una da 1/8 ne occuperà solo due e le

note da 1/16 verranno inserite, di fila, l'una sotto l'altra.

Quando SoundTracker incontra una nota in una colonna, la invia alla voce corrispondente (da sinistra, le quattro colonne corrispondono alle voci 0, 1, 2 e 3) ed interrompe la nota che stava suonando precedentemente. In altre parole, se vogliamo che un campione venga suonato sino alla sua fine naturale, è necessario lasciare libera la voce sino a che la forma d'onda non è terminata (dovreste riuscirvi ad orecchio). Notate che, per risparmiare memoria, molti strumenti sono digitalizzati solo nella loro parte iniziale e che quindi non sentirete il suono cadere naturalmente ma solo un... vuoto improvviso.

Le righe vengono suonate con un intervallo di 6/50 di secondo l'una dall'altra, ma questo valore può essere facilmente modificato con un comando apposito (vedi dopo).

Le colonne si dividono, a loro volta, in note e numeri. Le note (la notazione è chiaramente quella anglosassone, ma non si può avere tutto, no?) occupano i primi due spazi, il terzo è un numero che indica l'ottava a cui appartiene la nota (ve ne sono tre), gli altri quattro numeri hanno i seguenti significati:

1. Il numero di strumento usato (espresso in esadecimale). Se (come nel SoundTracker 2.3) sono presenti cinque numeri, i primi due sono il numero di strumento usato (sempre espresso in esadecimale).

2. Il numero di comando (vedi dopo).

3 e **4**. I parametri del comando dato (tanto per cambiare, sono in esadecimale).

Per inserire le note desiderate alla distanza voluta, scegliete l'opzione **Edit** e selezionate il pattern clickando con il mouse sul numerino di due cifre, che si trova in una casella posta sul pattern, vicino alla scritta "**Disk Status**".

Quindi, con l'aiuto dei tasti **cursore**, di **Del** e degli altri tasti (disposti come su un pianoforte, vedi la tabella 2 per l'elenco completo) potrete editare il vostro pattern ed ascoltarlo con l'opzione **Pattern**. Per caricare gli strumenti presenti sui vostri dischi ST, dovete utilizzare l'opzione **Preset** per sceglierli e quindi **Use Pset** per caricarli. Per modificarli potete sovrascriverli con altri.

Per caricare più di uno strumento, dovete agire sull'opzione **Sample** e selezionare il numero di strumento da caricare. Sotto di essa sono presenti altre due opzioni, **Repeat** e **Replen**. La prima permette di ripetere lo strumento per quante volte vorrete (Repeat = 2: nessuna ripetizione). Replen permette di ripeterne solo una parte.

E' bene modificare i parametri per tentativi per evitare suoni più simili ad una zanzara ubriaca che ad uno strumento della Filarmonica di Berlino.

Nel PresetEditor è possibile (anzi necessario) impostare i valori Repeat e Replen per i (pochi) strumenti che li richiedono. Alcuni SoundTracker richiedono anche un parametro **Volume** per ogni sample, altri si accontentano di deciderlo voce per voce. La principale differenza tra le varie versioni di SoundTracker, oltre nelle possibilità offerte all'utente (alcuni includono persino un digitalizzatore!), sta nel set di comandi riconosciuti e accettati.

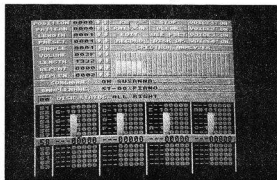
Solo cinque di essi sono standard; gli altri variano da versione a versione ed includono modulazioni di volume e di ampiezza, sliding del volume, fine anticipata del pattern, salto ad un'altra posizione, controllo del filtro e persino controllo di un altro programma creato dall'utente.

I comandi standard sono invece:

Arpeggio ($0). Serve per simulare gli accordi. Di fatto, l'altezza della nota corrente viene prima incrementata di tanti semitoni quanto vale la prima cifra dei parametri, poi lo stesso viene fatto con la seconda cifra, quindi la nota viene reimpostata al suo valore originale per poi ripetere di nuovo l'intero processo. Se, però, i parametri valgono 0, la nota viene lasciata inalterata; se non volete alcun effetto, impostate quindi 0 come comando e 00 come parametri. Attenzione, però: se volete che l'arpeggio valga per tutta la durata di una nota, dovete inserire la sequenza di comando anche nelle caselle sottostanti in quanto, ovviamente, una sequenza "0 00" (che è presente per default) elimina l'arpeggio selezionato.

Portamento su/giù ($1/$2). Permette di modificare la velocità di lettura del campione di tanti semitoni quanto è il valore del parametro; serve di fatto per "tirar su" o "tirar giù" una nota in maniera continua e senza doverne suonare altre.

Volume ($C). Imposta il volume di una voce ed è valido solo per la nota a cui si riferisce; ad esempio, per avere uno strumento a volume **$20** è necessario o farlo **sempre** seguire dalla sequenza **C20** o impostare il parametro Vol a 20.

Velocità ($F). Imposta il tempo, misurato in cinquantesimi di secondo, che intercorre tra la lettura di una riga del pattern e la successiva. Per ottenere una velocità di 5/50, invece di 6/50, è neces-

Glossario dei termini tecnici

Per facilitare anche ai non addetti ai lavori la comprensione di quanto contenuto in questo articolo, abbiamo pensato di inserire un breve glossario.

ADSR: sistema di "costruzione" dei suoni utilizzato in alcune tastiere e su alcuni computers (**C-64**); permette di definire un timbro con pochi valori, ma i suoni ottenuti non reggono il confronto di quelli "naturali".

Campionatore: è un circuito elettronico che trasforma, un suono, dalla sua forma analogica (ad es. voce, telefono, suono registrato su naschi e dischi) in un insieme di numeri atto ad essere elaborato da un computer (suoni su Amiga, compact disk).

Campione: è il risultato del lavoro del campionatore; su Amiga, è un file che contiene un suono.

Convertitore D/A: è il circuito elettronico "duale" del campionatore; trasforma un insieme di numeri (digitali) in una corrente elettrica (analogica), che, opportunamente amplificata ed inviata ad un altoparlante, si trasforma in un suono udibile. La risoluzione, cioè la capacità di un convertitore di riprodurre livelli diversi di intensità, si misura in bit: 4 bit significano 16 livelli diversi (come sul C/64); 8 bit 256 livelli (come sull'Amiga); 12 bit sono ben 4096 livelli (come su molti CD).

Filtro: su Amiga viene montato un filtro che attenua gradatamente tutte le frequenze superiori ad un certo numero di hertz. Può essere disinserito via software, e ciò causa suoni più metallici, ma meglio udibili. Il suo inserimento / disinserimento viene segnalato dal Led power (quello rosso): Led acceso = filtrò inserito, Led spento (o a luminosità dimezzata) = filtro disinserito.

Midi: è lo standard di comunicazione tra stumenti musicali elettronici.
La maggior parte delle tastiere oggi disponibili in commercio è, appunto, Midi-compatibile.

Pattern: è il "mattone" che serve a costruire un brano musicale su Amiga; rappresenta una sequenza di note (inseribili a piacere) eseguita dai programmi di musica.

Sample: vedi *Campione.*

Semitono: è la dodicesima parte di un'ottava (vedi tab. 2)

Sliding: è lo "scivolio" continuo o di una nota o del volume verso l'alto o verso il basso.

Strumento: è un campione caricato in memoria da un programma di musica e pronto per essere utilizzato in un brano.

Tremolo: è un effetto particolare in cui l'ampiezza di una voce viene modificata con frequenza di 5/10 hertz durante la sua esecuzione.
E' tipico, ad esempio, degli organi (elettronici e non); il suo effetto è certamente conosciuto dai nostri lettori.

sario che la prima riga del primo pattern da eseguire riporti la sequenza **F05**. Può essere comunque usato in qualsiasi posizione nel pattern, per quanto la soluzione più razionale sia sempre metterlo all'inizio. Per salvare i vostri capolavori, è necessario utilizzare **Disk Op**, che permetterà di scegliere se salvare o caricare un modulo o una song.

Il **modulo** è la **song** salvata assieme a tutti i suoi **strumenti**, e tende ad essere (molto) lungo. Perciò, è sempre meglio salvare in modo song.

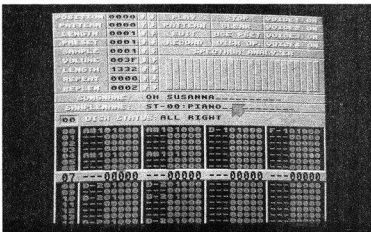
SoundTracker è un programma piuttosto permaloso, ed esige non solo di esser presente proprio sul disco di nome **ST-00**, ma anche che siano ivi presenti due subdirectories chiamate, rispettivamente, **Songs** e **Modules**; in caso contrario non funzionerà. Ricordate che i moduli sono sempre preceduti dal prefisso "**Mod.**", e che per utilizzare una musica nei vostri programmi è necessario caricarla in formato modulo.

SoundTracker, purtroppo, si "impalla" piuttosto facilmente senza degnare l'utente di una misera spiegazione.

Inoltre, oltre che dalla vostra abilità ed orecchio musicale, il risultato che si ottiene dipende moltissimo dalla qualità dei suoni che impiegate, che a sua volta dipende dall'hardware usato in fase di campionamento e dalla purezza del segnale analogico: se i suoni campionati direttamente da tastiere come la Korg e la Yamaha, sono veramente stupendi e cristallini.

Non alrettanto si può dire di una frase piena di sssibilanti, campionata da un vecchio disco con un campionatore da quindicimila lire....

POSTAMIGA

(a cura di Domenico Pavone)

Assembly e librerie

Ho cercato di seguire la serie di articoli dedicati al linguaggio macchina (in particolare "Prendiamoci una pausa" del n. 71 e "Dorage e la cornacchia" sul n. 76) ma mi sono scontrato con una serie di istruzioni per me incomprensibili, e che non sono spiegate in alcun modo. Che cosa diavolo è Execbase, e a che serve? Che sono OpenLibrary e CloseLibrary?...

(Paolo Maroncelli - Ravenna)

La difficoltà è più che comprensibile. Il linguaggio macchina non è certo uno scherzo, e la serie di articoli citata si rivolge a chi almeno i primi rudimenti li abbia già acquisiti.

L'argomento **Librerie**, tra l'altro, è di basilare importanza per qualunque programmazione di un certo livello, non solo in **Assembly**.

Tuttavia, poichè questo linguaggio necessita di particolari procedure per accedervi, vediamo di chiarire una volta per tutte i dubbi che assillano il nostro lettore, e sicuramente non solo lui.

Intanto, e questo è stato più volte descritto anche se non specificamente in rapporto all'Assembly, le cosiddette Librerie altro non sono che dei **gruppi di routines**, suddivisi a seconda dei compiti cui sono demandate.

Ognuno di questi "raggruppamenti" ha un suo nome: così se p. es. si vuole tracciare una linea sullo schermo, si ricorrerà alla libreria di nome **Graphics**; per aprire un file su disco si adopererà la libreria **Dos**, e così via. Nella tabella pubblicata a parte sono elencati i nomi delle principali librerie di sistema disponibili, che possono trovarsi tanto su **Rom** che su **disco**, per poi essere trasferite in **Ram**. Ma è quest'ultimo un aspetto che poco (in questa sede) interessa, anche perchè, normalmente, non è necessario il nostro intervento per segnalare al sistema dove reperire la libreria.

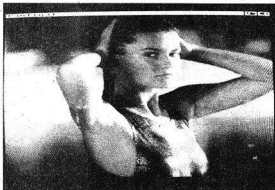
Ogni libreria, come ovvio, dispone di varie **funzioni**, ovvero di routines che svolgono una precisa mansione in rapporto ai dati che vengono loro inviati (quando e se necessari). Per accedervi, è inoltre disponibile, per ogni libreria, una **Jump Table** (= tabella di salti), automaticamente caricata in Ram da Amiga stessa: anche di questo, dunque, non ci dobbiamo occupare.

La Jump Table, in pratica, è una serie di istruzioni di salto poste una di seguito all'altra, ognuna diretta *verso* una precisa funzione di libreria. Il concetto, per chi ha avuto precedenti esperienze computerecce sugli 8 bit Commodore (leggi C/64 e C/128) non è nuovo: per adoperare una routine di sistema, piuttosto che "saltare" (con **Jsr**) all'indirizzo di memoria in cui questa è effettivamente memorizzata, si dirige il Jsr (**J**ump to **S**ub**R**outine) verso l'indirizzo della jump table contenente l'istruzione di salto alla routine desiderata.

Questo, se nei sistemi prima accennati era solo consigliabile, considerata la "stabilità" nel tempo degli stessi, con Amiga diventa una prassi obbligatoria. Una stessa routine, per esempio, può iniziare in due locazioni diverse, a seconda che si disponga della versione **1.2** oppure della **1.3**. Nella Jump Table, però, il salto a quella routine si troverà sempre al posto "giusto", e quindi risulterà valido indipendentemente da variazioni, anche future, del sistema operativo.

Il tutto, in effetti, è ancora più complesso di quanto si è fin qui accennato, ma per accedere alle funzioni di libreria in Assembly, è in pratica sufficiente effettuare un Jsr all'indirizzo della Jump Table nel quale è effettivamente contenuto il salto alla routine desiderata. Tanto per cambiare, con il **multitasking** sorge però un problema: dove trovare questa benedetta Jump Table, che può di volta in volta essere posizionata in locazioni diverse, dal momento che Amiga la "stiperà" nel primo spazio libero che trova in memoria.

Ed eccoci così giunti a quanto assilla il nostro lettore.

OpenLibrary è una funzione della libreria **Exec**, e serve (tra le altre cose) per accedere alla libreria desiderata, ed ottenere l'indirizzo di base della Jump Table ad essa associata. Con **CloseLibrary** si

ottiene invece (generalizzando alquanto) il rilascio della memoria impegnata dalla libreria, "chiudendola" definitivamente.

Prima di accennare a come funziona OpenLibrary, abbiamo però un'apparente contraddizione logica da risolvere.

Si è detto che OpenLibrary è una funzione della libreria Exec, e quindi anch'essa accessibile solo dopo che si è ricavato l'indirizzo della Jump Table di Exec.

Il circolo diventa decisamente vizioso: come aprire la libreria che consente di aprire le librerie?

Ebbene, l'unico indirizzo che potremmo definire *fisso* nella Ram di Amiga, è il puntatore alla base della Jump Table della libreria Exec: **la locazione numero** 4. L'indirizzo, naturalmente espresso come **Long Word** (4 byte, quindi occupante fisicamente le singole locazioni 4, 5, 6 e 7), viene per convenzione associato ad una costante di nome **Execbase**, o talvolta anche **Sysbase**. Ed ecco risolto uno dei dubbi della missiva.

Dopo tanta teoria, vediamo ora come adoperare OpenLibrary dando un'occhiata al disassemblato di queste pagine.

Si tratta di un comando che potremo chiamare **CLS**, sullo stile di quanto già esistente in ambiente Ms-Dos, che, una volta assemblata la elementare routine, provocherà uno svuotamento dello schermo ogni volta che lo si richiama da Shell (o Cli), o (più comodo) dall'interno di un batch file.

Il disassemblato, così com'è, è compatibile con il pacchetto **DevPac 2.14**. Chi volesse adoperare il Seka Assembler, dovrà sostituire "ALIGN 4" alle istruzioni "CNOP 0,4", ed aggiungere il simbolo due punti a tutti i nomi di etichette (uscita:, dos:, cls:) e alle costanti nella fase di inizializzazione (p. es. ExecBase: equ 4, Write: equ -48, eccetera). Le varie istruzioni possono invece rimanere tali e quali per entrambi gli assemblatori.

Chi non possedesse un assembler, o non sapesse comunque adoperarlo correttamente, può ricorrere al "caricatore" AmigaBasic, anch'esso pubblicato in queste pagine. Il suo uso è semplicissimo: basta caricare Amigabasic e copiare il listato, prestando una certa attenzione alla sfilza di numeri inseriti nelle righe Data.

Una volta mandato in esecuzione, provvede a controllare se non siano presenti valori incongruenti, eventualmente segnalandoli a video. Per avere su disco il comando Cls già bello e pronto, occorre rispondere ad un input che chiede di specificare in quale unità memorizzare il file. E' importante concludere il nome del disco o della periferica con il simbolo due punti(:). Volendo, si può anche precisare una subdirectory (p. es. **Disco:miecose/**), in questo caso inserendo obbligatoriamente, come ultimo carattere, la barra obliqua.

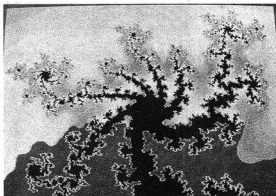
Niente altro. A fine operazioni, si potrà attivare Cls da Shell o Cli secondo le consuete modalità operative del Dos.

Ma torniamo al disassemblato.

Nella prima riga, si può notare subito l'assegnazione

```
*------------------------------------------------
*           COMANDO CLS (CLear Screen)
*------------------------------------------------
ExecBase        equ 4
OpenLibrary     equ -552
CloseLibrary    equ -414
Output          equ -60
Write           equ -48
*------------------------------------------------
        move.l  #Dos,a1          ;Nome Libreria
        moveq   #0,d0            ;Qalunque versione
        move.l  ExecBase,a6      ;Locazione 4 in A6
        jsr     OpenLibrary(a6)  ;Apre libreria Dos
*------------------------------------------------
        tst.l   d0          ;Controlla se tutto ok.
        beq     uscita      ;Se D0=0, errore.
        move.l  d0,d4       ;Base Jump Table in D4.
*------------------------------------------------
        move.l  d4,a6        ;Attiva funzione Output
        jsr     Output(a6)   ;della libreria Dos.
*------------------------------------------------
        move.l  d0,d1     ;Handle di output per Write
        move.l  #cls,d2   ;Mette valore 12 in D2.
        move.l  #1,d3     ;Lunghezza = 1 byte.
        move.l  d4,a6     ;Attiva funzione Write
        jsr     Write(a6) ;della libreria Dos.
*------------------------------------------------
        move.l  d4,a1             ;Base di Dos in A1
        move.l  ExecBase,a6       ;Locazione 4 in A6
        jsr     CloseLibrary(a6)  ;Chiude libreria
*------------------------------------------------
uscita  rts                       ;Ritorno a Shell
*------------------------------------------------
Dos     dc.b    'dos.library',0  ;Nome libreria
        cnop    0,4              ;Allineamento a long
cls     dc.b    12               ;Ascii di Form Feed
        cnop    0,4              ;Allineamento a long
```

Nome	Compiti
Dos	amigados, inpout/output, gestione files.
Exec	accesso librerie, allocazione memoria, ecc.
Graphics	funzioni grafiche in genere e sprite.
Icon	gestione icone di Workbench.
Intuition	menu, gadget, schermi, finestre, ecc.
Mathffp	funzioni aritmetiche (fast) a 32 bit.
Mathtrans	funzioni trascendentali a 32 bit (seno, coseno, ecc).
Mathieedoubbas	funzioni aritmetiche a 64 bit.
Potgo	funzioni per porte giochi.
Timer	lettura e settaggio orologio interno.
Translator	gestione sintetizzatore vocale.
Diskfont	caratteri diversi dal default interno.

Librerie Amiga di uso piu' comune

della costante Execbase che, in pratica, corrisponde sempre a 4, da intendere come la long word "contenuta" nelle 4 locazioni, dalla quarta in poi. Per inciso, i numeri non preceduti dal simbolo "$" sono da intendersi in notazione decimale.

Subito a seguire, si nota una serie di assegnazioni, riferite a nomi di funzioni di libreria, tutte con un segno negativo. Eccoci al punto cruciale della nostra dissertazione. I valori negativi indicano lo "scostamento" necessario, rispetto alla base della jump table, per raggiungere la funzione desiderata.

Più in concreto: supponendo che sia **X** l'indirizzo di base della tabella dei salti della Dos Library, che corrisponde fisicamente alla locazione più alta (è memorizzata per così dire alla rovescia), l'entry che "punta" all'inizio della routine **Write** (scrive qualcosa sullo schermo) si troverà esattamente alla locazione **X - 48**.

Per ricavare tali valori, non c'è altra strada che consultare una documentazione specifica come i Rom Kernel Manual... almeno se si adoperano assemblatori "stringati" come il Seka. I pacchetti più evoluti, infatti, hanno in dotazione i cosiddetti **Include**, per cui può non risultare necessario inizializzare le costanti. Nel disassemblato, per chiarezza, si è ricorsi alla forma più esplicita, anche se il DevPac consentirebbe di semplificare le chiamate a funzioni di libreria.

Passando al codice vero e proprio, le prime istruzioni provvedono ad aprire la libreria Dos.

In pratica, è l'istruzione **Jsr OpenLibrary(a6)** a svolgere questo compito. Tradotta in termini logici, significa "salta all'indirizzo che si ottiene sommando il contenuto del registro **A6** alla costante OpenLibrary. Essendo quest'ultima negativa, si avrà in pratica una sottrazione.

A quell'indirizzo, nell'ambito della Jump Table della libreria Exec, ci sarà una ulteriore istruzione di salto verso la routine OpenLibrary vera e propria.

Il registro A6 era stato, subito prima del salto, "riempito" con il contenuto della locazione 4, come già detto corrispondente all'indirizzo di inizio della tabella di salti di Exec.

Alla funzione OpenLibrary, per funzionare a dovere, va inviato come parametro il nome della libreria da aprire (in questo caso Dos.Library) fatto concludere da uno zero. Più specificamente, va inserito (prima del salto) l'indirizzo ove è memorizzata la stringa "Dos.Library" nel registro A1, mentre D0 viene azzerato, per indicare che qualunque versione della libreria va bene.

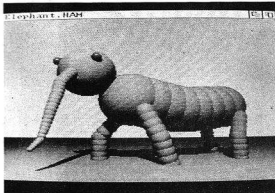
Dopo il Jsr, il registro **D0** conterrà l'indirizzo utile per adoperare la libreria **Dos**, ovvero l'indirizzo di base della sua jump table, a meno che non siano sorti dei problemi: in quest'ultimo caso, D0 conterrà zero.

Ora, tutto si fa più semplice(!). Seguendo la stessa procedura, ma inserendo in A6 la nuova **base** di libreria (si segua il disassemblato), si potrà adoperare la funzione **Output** della Dos.library per ottenere (sempre in D0) un "Handle" necessario poi alla funzione **Write** per inviare alla finestra attiva un carattere Ascii **12**, detto Form Feed, che provoca la cancellazione di quanto presente nella finestra.

Si vedano i commenti del disassemblato per le inizializzazioni dei registri.

Infine, viene ripristinato in A6 (va SEMPRE adoperato questo registro per le chiamate a funzioni di libreria) l'indirizzo di ExecBase, ed invocata CloseLibrary con A1 contenente la base della Dos Library: in pratica, quest'ultima verrà così chiusa.

L'argomento va ovviamente approfondito, ma ce n'è ab-

```
PRINT "Memorizzazione su disco del comando CLS
PRINT "per chi non dispone di un assemblatore"
PRINT"-----------------------------------"
PRINT
FOR x=1 TO 144:READ a: b=b+a :NEXT
IF b <> 6824 THEN
    PRINT "ERRORE NELLE LINEE DATA!": END
END IF
RESTORE
PRINT "In quale disco o unita' vuoi"
PRINT "memorizzare il comando Cls?"
PRINT "(p. es. Df0:, Df1:, Miodisco:"
PRINT "senza dimenticare i due punti)"
PRINT : INPUT dev$: file$ = dev$ + "CLS"
OPEN file$ FOR OUTPUT AS 1
FOR x=1 TO 144
    READ a:PRINT #1,CHR$(a);
NEXT
CLOSE: END
DATA 000,000,003,243,000,000,000,000,000,000
DATA 000,001,000,000,000,000,000,000,000,000
DATA 000,000,000,021,000,000,003,233,000,000
DATA 000,021,034,124,000,000,000,066,112,000
DATA 044,121,000,000,000,004,078,174,253,216
DATA 074,128,103,000,000,042,040,000,044,068
DATA 078,174,255,196,034,000,036,060,000,000
DATA 000,080,038,060,000,000,000,001,044,068
DATA 078,174,255,208,034,068,044,121,000,000
DATA 000,004,078,174,254,098,078,117,100,111
DATA 115,046,108,105,098,114,097,114,121,000
DATA 000,000,012,000,000,000,000,000,003,236
DATA 000,000,000,002,000,000,000,000,000,000
DATA 000,002,000,000,000,036,000,000,000,000
DATA 000,000,003,242
```

bastanza per cominciare a capire come muoversi tra funzioni, librerie, ed amigaglia varia.

✉

Se ci sei...

Vorrei sapere se c'è un modo per scoprire, in AmigaBasic se un file esiste o meno, come l'If Exists del Dos. Ci ho provato con Open, ma il contenuto del file è andato perduto.
(Mariano Gilardi)

Il modo c'è, anzi ce n'è più d'uno.
Un uso oculato dello statement **Open** può già risolvere il problema, ma con tutte le cautele del caso.

Per accertare la preesistenza o meno di un file, si può infatti tentare di aprirlo in Input, ed intercettare un eventuale errore **File Not Found** (error **53**) tramite la condizione **On Error** ammessa da AmigaBasic.

Questo tipo di procedura però, se non adoperato correttamente, può talvolta provocare l'azzeramento del contenuto del file (solo per un eventuale errore di programmazione, però...).

Comunque, per andare sul tranquillo, è più semplice adoperare una delle due tecniche illustrate nei due listati basic di queste pagine.

In entrambi, è in pratica implementata una SUB completamente autonoma, inseribile quindi in qualunque tipo di applicazione. Il listato 1, per raggiungere lo scopo, fa uso unicamente delle istruzioni del basic, mentre con il listato 2 si ricorre alla libreria Dos di sistema, seguendo una procedura che può risultare utile anche in linguaggi meno semplici del basic.

Esaminiamoli più da vicino.

Nel listato 1, la prima parte serve più che altro da esempio, in quanto si limita a chiedere in input il nome di un file del quale controllarne l'esistenza. Non si dimentichi, se il caso, che va inserito il percorso completo (per esempio **Df1:nomefile**).

Poi, viene richiamato il sottoprogramma, o funzione basic che dir si voglia, di nome **FileCtrl**, il quale richiede gli venga fornito il nome del file da esaminare, naturalmente sotto forma di stringa (**Nome$**). La chiamata della funzione, va effettuata con:
FileCtrl file$, flag

La seconda variabile, **Flag**, non necessita di alcuna inizializzazione, serve solo per ricevere "di ritorno" un valore che sarà **0** se il file non è stato trovato, ed **1** in caso di esito positivo nella ricerca.

Questa variabile, volendo, la si sarebbe potuta rendere **Shared** nella Sub, ovvero *condivisa* con il resto del programma, evitando così la necessità di doverla citare nella chiamata della funzione. Procedendo, però, come proposto nella breve routine, si rende la funzione COMPLETAMENTE autonoma, cosa che va ricercata quanto più possibile nel crearsi una propria libreria di funzioni.

Ma veniamo al cuore della routine. In pratica si limita a *tentare* di aprire il file **Nome$** (che nel programma principale è **File$**, o qualunque altro nome di variabile si sia adoperato), ma in modalità **Append**, non **Input**. Con ciò si ottiene che se il file esiste, non si rischia alcuna sua manomissione, mentre in caso contrario ne verrebbe creato uno ex novo.

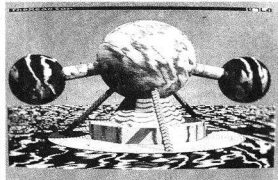
In entrambi i casi, la sua dimensione verrà per così dire "registrata" nello statement **LOF()**, che risulterà quindi uguale a zero se il file è stato appena creato (cioè non esisteva prima). Nel listato si è adoperato un numero di file **11** per evitare eventuali sovrapposizioni con altri file eventualmente aperti nel programma principale, ma si può anche adoperare **255**, numero massimo consentito.

A questo punto un banale **If** si incarica di controllare lo stato di Lof, azzerando o settando (=ponendo ad 1) la variabile **Fl**, che, una volta usciti dal sottoprogramma, corrisponderà in pratica alla già vista Flag.

Si noti come, se **Lof** è nullo, e quindi il file non esisteva, occorre, prima di tornare alla sezione principale, *cancellare* il file che Open ha momentaneamente creato. Il che significa prima chiuderlo (istruzione Close) e poi adoperare il comando **Kill**.

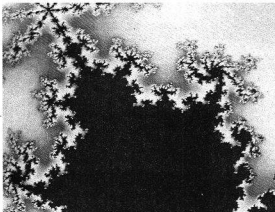
Nel caso il file fosse già presente, si può inoltre evitare di richiuderlo, qualora lo si dovesse manipolare nel programma chiamante (vedi **Rem** nel listato).

Il secondo listato segue l'identico algoritmo, solo che è richiesta, nella sezione "main", l'apertura della **Dos Library**. Senza troppo rimarcare un argomento trattato praticamente in ogni numero della rivista, si ricorda che all'uopo deve essere accessibile al programma il file **Dos.bmap**, presente nella directory **Basicdemos** del disco **Extras**.

Nel subprogramma, viene anzitutto aggiunto un **Chr$(0)** in coda alla stringa rappre-

```
REM            LISTATO 2
PRINT "Demo per funzione basic"
PRINT "che controlli l'esistenza"
PRINT "di un file ricorrendo"
PRINT "alla libreria Dos"
PRINT"------------------------"
PRINT
DECLARE FUNCTION Lock& LIBRARY
LIBRARY "dos.library"
PRINT"Nome completo del file ";
INPUT file$
Filectrl file$, flag&
IF flag&=0 THEN
    PRINT file$" non esiste!"
ELSE
    PRINT "il file esiste gia'"
END IF
END
            REM CONTROLLA ESISTENZA FILE
SUB Filectrl (nome$, fl&) STATIC
    nom2$=nome$+CHR$(0)
    nm&=SADD(nom2$)
    fl&=Lock&(nm&,-2)
    CALL Unlock(fl&)
END SUB
```

sentante il nome del file, come richiesto dal sistema nelle funzioni di libreria. Quindi si richiede il **Lock** del file, ovvero una specie di canale di accesso, che può essere esclusivo o meno a seconda che si adoperi **-1** oppure **-2** come secondo parametro della funzione Lock.
Nel nostro caso, adoperiamo -2, mentre come primo parametro forniremo l'indirizzo in memoria della stringa prima trattata, ottenuto tramite l'istruzione **Sadd**.

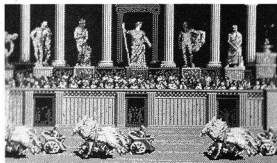
Si noti, inoltre, come la stringa originale (senza il Chr$(0) aggiunto) viene mantenuta integra, in modo che possa poi essere eventualmente visualizzata nella sezione principale del programma, senza lo sgradevole effetto del carattere anomalo in coda.

La funzione Lock restituisce infatti un valore 0 se non è possibile l'accesso ad un file di quel nome. Adoperando, per richiedere il Lock, la stessa variabile che verrà poi

restituita alla sezione principale del programma (**fl&**, che corrisponde nel progr. chiamante al solito Flag), non resta altro che chiamare la funzione **Unlock**, senza neanche la necessità di adoperare i vari If... Else visti per la tecnica precedente.

Si occuperà direttamente la sezione chiamante di verificare se Flag& è uguale a zero, e quindi se il file esiste o meno.

Il tutto in massima sicurezza, e con la possibilità di applicare quanto visto alle più disparate esigenze.

```
REM            LISTATO 1
PRINT "Demo per funzione basic"
PRINT "che controlli l'esistenza"
PRINT "di un file senza ricorrere"
PRINT "a librerie di sistema"
PRINT"------------------------"
PRINT
PRINT "Nome completo del file ";
INPUT file$
Filectrl file$, flag
IF flag=0 THEN
    PRINT file$" non esiste!"
ELSE
    PRINT "il file esiste gia'"
END IF
END
            REM CONTROLLA ESISTENZA FILE
SUB Filectrl (nome$, fl) STATIC
    OPEN "a",#11,nome$
    IF LOF(11)=0 THEN
        fl=0
        CLOSE #11:REM non eliminabile
        KILL nome$
    ELSE
        fl=1
        CLOSE #11:REM eliminabile
    END IF
END SUB
```

Troppo lenti? dipende...

Ho copiato il batch file da voi pubblicato in Amigafacile (settembre '90), che controlla se si può scrivere in un disco. Funziona, però ho notato che è piuttosto lento, come anche molti altri che si trovano nella stessa rubrica. Come mai?
(Roberto ?? - Firenze)

Il problema, purtroppo, esiste. Non è però legato al particolare file batch, ma a qualunque file di tipo **Script** di una certa entità.

Un batch file, infatti, non fa altro che provocare l'esecuzione di un gruppo di *comandi*, che poi non sono altro che normali *programmi*.

Si consideri, come esempio, un batch file contenente due banali istruzioni:

Makedir Ram:Util
Echo "Directory creata"

Il sistema, in pratica, dovrà: 1) Cercare nella directory corrente se è presente il file di nome **Makedir**.

2) Se non lo trova, scorrere la directory **C** per cercare di rintracciarlo al suo interno.

3) Se non lo trova neanche lì, scorrere eventuali altre directory inserite nel Path.

4) Una volta trovatolo, caricarlo in memoria e mandarlo in esecuzione.

Stessa trafila, naturalmente, viene poi ripetuta per il comando **Echo**, e per ognuno degli eventuali altri inseriti in un batch.

Se a tutto questo si aggiunge, poi, la comune frammentazione dei files in un dischetto, il panorama dei rallentamenti possibili si fa completo. Tutti (o quasi) i comandi del Dos di Amiga, come è noto, sono infatti **esterni** e non **interni** (come sono quelli, ad esempio, del sistema operativo Ms-Dos); ciò significa che, per essere eseguiti, devono esser caricati da una memoria di massa esterna.

Qualche rimedio, però, c'è, anche se l'unica **vera** soluzione sarebbe quella di acquistare un hard disk.

Intanto, un primo provvedimento può riguardare lo stesso floppy in cui sono memorizzati i comandi, il cui accesso ai files può essere notevolmente sveltito ottimizzandone la "geografia" interna. In pratica, si tratta di adoperare programmi come Disk Arranger, Bad, D.Optimizer, eccetera.

Un'altra "limata" ai rallentamenti può essere data evitando la ricerca dei comandi in più directory. Se, infatti, nel batch file viene precisato il percorso di ogni comando, il sistema andrà a colpo sicuro nella directory che li contiene, senza effettuare la ricerca in quella corrente, che quasi mai corrisponde alla C. In pratica, si tratta di trasformare il batch di esempio prima visto in modo che risulti così conformato:

C:makedir Ram:Util
C:echo "Directory creata"

Si noti come, adoperando la descrizione del device logico **C:** (e **non** C/), si è certi che la directory C di sistema venga in ogni caso raggiunta, indipendentemente dalla posizione corrente.

Infine, se proprio si vuol andare "a scheggia", al trucco appena visto si può associare (o adoperare anche con batch senza indicazioni di percorso) il trasferimento di tutti i comandi del Dos in Ram Disk, assegnando poi C: a questo device.

In pratica, si tratta di eseguire una istruzione...

Copy C: Ram:

...in modo diretto da Shell, o inserirla nella startup - sequence. In tal modo tutti i files della directory C di sistema verranno copiati nella Ram Disk. A questo punto, non resta che impartire un...

Assign C: Ram:

...e tutti i comandi verranno caricati a velocità supersonica, direttamente dalla Ram Disk.

La copia di tutti i files può richiedere un certo tempo, ma il lavoro successivo ne risulterà ampiamente snellito. Inutile aggiungere che, ad evitare improvvisi black out a cura del caro Guru, con relativa perdita del contenuto della Ram disk, l'ideale sarebbe adoperare piuttosto la **Rad**, resistente al reset, che non obbligherebbe a ripetere la (lunga) procedura di copia dei comandi.

Amiga 500 "tiene"?

Sto per acquistare un Amiga, e sarei orientato sul modello 500 più che altro per motivi economici, pensando di espanderlo pian piano. Mi è stato detto, però, che aggiungendo espansioni, periferiche, eccetera, l'alimentatore non può "reggere" da un punto di vista elettrico.
(Francesco De Carli - Milano)

Effettivamente può insorgere qualche problema se proprio si "carica" troppo il 500, modello per il quale è stata, sì, prevista una certa espandibilità (intesa in generale, non solo in rapporto alla memoria), ma non certo come sui 2000, fabbricati prevedendo ampliamenti delle già notevoli potenzialità di base.

C'è da dire che, vista la considerevole disponibilità di accessori e "upgrade" hardware per A500, tale da avvicinarlo sempre di più alle prestazioni del modello superiore, qualcosa sta venendo fuori per sopperire al problema dell'eventuale sovraccarico elettrico.

Giusto per fare un esempio, è stato da poco commercializzato un nuovo alimentatore per Amiga 500 della **Cabletronic**, importato da uno dei nostri inserzionisti (**Flopperia**), che regge un carico di 0,5 Ampere, a fronte degli 0,3 degli alimentatori comunemente in dotazione alla macchina.

Pur senza attribuire ad Amiga 500 la stessa propensione professionale dei modelli superiori, si può insomma tranquillamente pensare di acquistarlo in previsione di una configurazione "medio/minima", senza eccessivi timori per future mirabolanti migliorie.

Problemi di modem

Ho alcuni problemi con il mio modem: non so scompattare i file ricevuti con programmi come Lharc; inoltre non so come far partire i programmi basic prelevati sempre via modem.
(Lawrence Iviani - Trieste)

Prima di rispondere, è il caso ricordare che di qualunque programma, AmigaBasic compreso, va prima di ogni altra cosa consultato il manuale, quantomeno a livello di "rapida scorsa".

Anche quando non è sufficientemente chiaro.

Non è questo, però, il caso di **Lharc**, il compattatore / scompattatore per Amiga "made in Italy" (l'autore è **Paolo Zibetti**), distribuito con un esauriente manuale di istruzioni.

Qualche problema, per chi è proprio agli inizi, può sorgere unicamente in rapporto al "dove", più che al "come" i files vengono scompattati.

Supponiamo di avere lanciato il sistema con il disco Workbench, e di avere in un altro disco, di nome **Down**, il file con suffisso **.Lzh** appena ricevuto via modem.

Anzitutto, è importante sapere dove si trova il programma **Lharc**.

Se si dispone di spazio sufficiente, è più comodo tenerlo nella directory **C** del disco di lancio, altrimenti, al momento di attivarlo, è indispensabile specificarne l'intero percorso. Prendiamo, dunque, in esame il caso più complicato: Lharc è memorizzato nella **Root** directory di un terzo dischetto, per esempio di nome **Util**.

Intanto, come illustrato dall'help che appare se si attiva Lharc senza specificare alcun parametro, la sua sintassi normale per scompattare un file è...

Lharc X file

...con **file** che indica il nome del file da scompattare, del quale **non** è necessario precisare il suffisso (.lzh), e con **X** che può essere sostituito da **E**.

Adoperando direttamente questa forma (come ovvio sostituendo a "file" il nome del file da scompattare), è però indispensabile, anzitutto, che Lharc sia *direttamente accessibile*, ovvero che si trovi nella directory *corrente*, oppure nella directory C del disco di boot, oppure in una directory inserita nel Path di ricerca (si veda descrizione del comando in **Amigafacile** n. 75). In caso contrario, tornando al caso prima prospettato, occorre specificare il "percorso" del programma, che risulterebbe nell'esempio...

Util:Lharc.

A meno che non si trovi nella directory corrente, lo stesso discorso vale per il file ".lzh": o ci si sposta nella directory in cui si trova (tramite il comando **CD**), oppure se ne deve precisare il percorso. Nel nostro caso, si avrebbe in definitiva un comando...

Util:Lharc X Down:file

...con il nome dei dischetti eventualmente sostituibile con la specifica di periferica (df0:, df1:, Ram:, Dh0:).

I file scompattati, di norma, saranno memorizzati nella directory *corrente*. Questo può essere un effetto indesiderato, ma è molto facile stabilire dove "far finire" i files scompattati. Anzitutto, banalmente, ci si può portare nella directoy voluta adoperando CD. Se, per esempio, volessimo i files nello stesso dischetto contenente l'archivio **.lzh**, basterebbe un...

Cd Down:

...prima di impartire l'istruzione prima descritta.

A semplificare ulteriormente le cose, lo stesso Lharc prevede la possibilità di dirottare i files scompattati verso una directory diversa da quella corrente.

Basta aggiungere, in coda alla sintassi prima vista, il **path** di destinazione.

Quindi, se per esempio volessimo trovare in **Ram Disk** i file "sputati fuori" da Lharc, l'esempio prima visto diventerebbe...

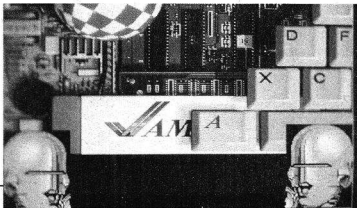
Util:lharc X down:file Ram:

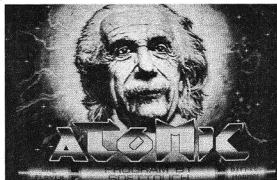
Unico elemento cui prestare attenzione è l'eventuale presenza di subdirectory nel percorso di destinazione.

In questo caso è necessario concludere la stringa descrittiva con una barra obliqua. Sempre rifacendosi allo stesso esempio, se si vogliono memorizzare i file scompattati nella directory **Scomp** presente in Ram disk, il comando da digitare risulterebbe...

Util:lharc X down:file Ram:scomp/

Quanto ai programmi basic "downloadati" via modem, per utilizzarli non c'è che da caricare l'interprete Amigabasic, biclickando la sua icona da workbench (non sarà necessario specificare che si trova sul disco Extras, vero?), e poi adoperare l'opzione Open del menu "Project", inserendo, nel successivo requester, il nome del file basic, tanto per cambiare completo di percorso (ormai dovrebbe essere chiaro a che cosa ci si riferisce).

Un bel Run impartito direttamente nella finestra di output, oppure "Start" dal menu Run, ed il gioco è fatto.

Comandi DOS gia' pubblicati

n.75	N.76	N.77	N.78
	Caratteri		
Assign	Speciali	Execute	Cd
Copy	Delete	Direttive	Ed
Date	Format	Batch	Break
Dir	Protect	If	
Install	Rename	Skip...lab	
Path		Quit	
Search			
Sort			

NOTA BENE
Le parole chiave di ogni comando sono rappresentate in maiuscolo e vanno (eventualmente) adoperate così come sono.
In minuscolo, sono invece riprodotti i parametri che vanno ridefiniti dall'utente.

Which

In quasi tutte le applicazioni che riguardano i comandi di AmigaDos, il problema più rilevante, per i non esperti, è rappresentato dal *percorso* legato alla citazione di un particolare file.
Con Which, una volta tanto, il problema è ribaltato. Questo comando infatti, presente solo dalla versione **1.3** in poi, consente proprio di rintracciare la directory ove è memorizzato un file, o verificarne l'esistenza. Purchè, però, questo sia presente nell'ambito del path di sistema (vedi descrizione della sintassi).
Which può spesso risultare utile impartito in modo diretto, ma trova tuttavia una valida applicazione anche nell'ambito di batch files, grazie alla sua capacità di modificare lo stato della condizione **Warn**.

WHICH

Un esempio

Nella descrizione della sintassi del comando, si fa cenno all'azione di **Which** nei confronti della condizione **Warn**, utile al fine di eventuali diramazioni nell'ambito di un batch file.
Ebbene, la condizione Warn viene impostata a **vera** se il file precisato nella sintassi di Which **non** viene trovato. Quindi, in poche parole, inserendo un **If Warn** in un file comandi, si testerà la **non esistenza** del file nel percorso di ricerca di sistema, mentre **If Not Warn** ne verificherà il ritrovamento.
Come esempio pratico, si consideri questo breve batch, che chiameremo **GO**, in considerazione dello scopo che si prefigge: facilitare la frequente necessità di portarsi nella directory ove è contenuto un certo file.

```
.key file
if <file>X eq X
echo "SINTASSI: Go nomefile"
quit
endif
which <file> nores
if not warn
echo ""
echo "Digita il percorso"
cd >nil: ?
endif
```

Una volta copiatolo con **ED** (o altro editor ascii), va mandato in esecuzione con **Execute Go Nomefile**, o semplicemente **Go Nomefile** se si provvede a settare il bit Script del file (comando: **Protect Go +S**).
Nomefile, come ovvio, precisa il file da ricercare. Il batch controlla che questo parametro venga effettivamente inserito (**If...Eq**. Si veda Amigafacile n. 77), quindi provvede ad implementarne la ricerca tramite Which. Si noti l'uso dell'opzione **Nores**, per escludere la lista dei comandi residenti (non è certo possibile usare **Cd** per la lista residente!).
Se il file viene trovato (Not Warn), il suo percorso verrà visualizzato sul monitor dal comando Which.
A questo punto un comando Cd (rediretto verso **Nil:**, cioè a vuoto, per evitarne l'output su schermo), accompagnato dal **template** (l'uso del punto interrogativo) consentirà che si copi il percorso appena visto (senza il nome del file) ed immediatamente ci si troverà nella directory voluta.

Which

Impartito senza alcun parametro non sortisce altro effetto che una segnalazione di errore "Bad args". Naturalmente, come del resto tutti i comandi Dos inseriti nel disco di sistema, è fornito del suo bravo "template", ovvero il mini-help che viene attivato se si inserisce un punto interrogativo come unico parametro.
Come dovrebbe esser noto, in questo caso viene visualizzata (in maniera non troppo esplicita, per la verità) una stringata e simbolica descrizione della sintassi accettata dal comando, mentre il computer si pone in stato di attesa.
Digitando a questo punto i parametri da attribuire al comando, e facendo seguire il tutto dall'ovvia pressione del Return, questi verranno accettati esattamente come se fossero stati fatti seguire a Which (o all'istruzione che si sta adoperando) nella riga di comando Shell (o Cli).

Nores

Al contrario di **Res**, questa keyword esclude la lista residente dalla ricerca effettuata da Which. L'uso sintattico di una forma comprensiva di Nores, può risultare utile più spesso di quanto si creda, considerato il fatto che i comandi residenti vengono scanditi per primi da Which, impedendo in certi casi l'acquisizione di una informazione riguardante il disco.
Come esempio, e sempre considerando una finestra Shell aperta dal disco Workbench, si provi ad impartire **Which List**. In risposta, si otterrà RESIDENT List. Tutto regolare, ma se volessimo ipoteticamente sapere in quale directory del disco è memorizzato il comando, non potremmo esimerci dall'adoperare **Which List Nores.**
Anche in questo caso, comunque, l'utilità pratica è maggiormente legata alla stesura di eventuali batch, come quello illustrato nel riquadro riservato agli esempi pratici.

WHICH file RES NORES

File

Il nome del file da ricercare. Come ovvio senza alcuna specifica di percorso, visto che Which serve proprio per prenderne conoscenza. Oltre che file in senso stretto, questo parametro può anche indicare il nome di una directory.
Supponendo di aver lanciato il sistema con il disco Workbench in dotazione al computer (non modificato!), si provi ad aprire una finestra Shell biclickando nella sua icona, quindi a digitare **Which Format**. Dopo un breve intervallo, apparirà sullo schermo l'indicazione...
Workbench1.3:Sysyem/format
In pratica l'intero percorso del file Format. Impartendo, invece, un improbabile **Which For**, si otterrà in risposta un semplice "for not found", in quanto non esiste alcun file di nome **For** nel disco.
Attenzione, però: non tutto è così lineare come sembra.
Sappiamo, per esempio, che nella directory **Fonts** del disco esiste una directory di nome **Ruby**, uno dei caratteri alternativi al **Topaz** di default. Bene, si provi ad impartire **Which Ruby**. Nonostante l'esistenza di quel nome sia ampiamente verificabile con un banale **Dir Fonts**, in risposta si otterrà ugualmente una segnalazione di file non trovato.
Volendo, si può adoperare la forma **Which Fonts/Ruby**, peraltro decisamente poco pratica; se ne otterrebbe una semplice conferma di quanto già ben noto, ovvero che Ruby è presente all'interno della directory Fonts.
Il motivo di tutto ciò è presto detto: Which effettua la sua ricerca in un preciso ambito, che comprende nell'ordine:

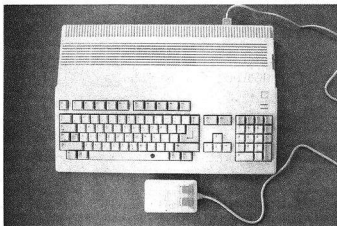
1) La lista dei comandi residenti.
2) La directory corrente.
3) La directory C.
4) Tutte le directory incluse nel percorso generale di ricerca, verificabile impartendo il comando **Path**.

Si provi, dunque, ad impartire Path nella finestra Shell. La directory Fonts non risulterà compresa nell'elenco mostrato a video, mentre lo sarà quella **System**, all'interno della quale è stato prima rintracciato il comando **Format**. Nel caso particolare del disco Workbench, tutte le directory comprese nel percorso di ricerca risultano tali in virtù della procedura di boot del disco, come palpapile impartendo **Type S/Startup-sequence** ed esaminando la riga comandi che inizia proprio con il comando Path.

Res

Aggiungendo questo parametro, si forza Which ad effettuare una ricerca solo tra i files resi residenti con il comando Resident.
Quindi, p. es., impartendo **Which Run** si otterrebbe in risposta... **Workbench1.3:c/run**, mentre **Which Run Res** darebbe un responso negativo, a meno che non si sia inserito **Run** tra i comandi residenti.
Chiaro che, a meno di non avere residenti una marea di files, l'utilità pratica di questo parametro non è molto elevata se impartito in modo diretto: basta infatti un normale **Resident** per ottenere più velocemente l'informazione voluta.
Può invece risultare (talvolta) utile nell'ambito di un batch file, per testare la presenza, o meno, di un file tra i comandi residenti.

I DEVICE

Con il termine **Device**, in ambiente AmigaDos, vengono specificati tutti quei **dispositivi** che, convenzionalmente, sono seguiti dal simbolo due punti(:).
In questa catogoria rientrano, quindi, le periferiche - disco come **Df0:**, **Df1:**, eccetera, ma anche la **Ram Disk** che, seppur in modo virtuale, emula la gestione di un drive. In aggiunta a queste periferiche fisiche, è inoltre nota l'ampia disponibilità di **pseudo-device**, ovvero i cosiddetti Device **Logici** (**C:**, **Devs:**, **Fonts:**, eccetera) di cui spesso ci siamo occupati in queste pagine (vedi soprattutto descrizione del comando **Assign** sul n. 75). Nella prima categoria, quella dei dispositivi fisici, rientra però anche una serie di device di importanza per nulla secondaria, dei quali ci occuperemo in questa sede.
Si sta parlando, in particolare, della gestione delle **porte** di comunicazione (**Ser:**, **Par:**, **Prt:**), e degli strumenti più immediati di interfacciamento con il computer, legati all'uso della tastiera (**Con:**, **Newcon:**, **Nil:**, **Raw:**).

SER:

Indica la porta di comunicazione **seriale**, attraverso la quale si può connettere Amiga ad un **modem**, ad un altro **computer**, o ad una **stampante** seriale. Da Amigados è possibile dirigere l'output verso uno dei dispositivi succitati, ma da un punto di vista pratico è da prendere in considerazione solo l'eventualità di adoperare una stampante.
La trasmissione di dati via modem, o anche quella rivolta ad un altro computer, comporta infatti una complessità che va al di là del semplice uso del dos (quantomeno nelle possibilità di input), affrontabile adeguatamente (e facilmente) tramite specifici programmi applicativi.
L'uso dell'interfaccia seriale richiede il settaggio preliminare della velocità di invio dei dati, in accordo con quella supportata dal dispositivo esterno. Tale **velocità**, espressa in **bps** (**b**it **p**er **s**econdo), va impostata tramite il programma **Preferences**, presente nella directory **Prefs** del disco Workbench, che presenta nella sua schermata di avvio l'opzione **Change Serial**. Clickandovi, si accede alle regolazioni della porta, in alternativa raggiungibili direttamente biclickando l'icona **serial** della stessa directory.
Per modificare la velocità di trasmissione, è sufficiente agire sulle freccette poste lateralmente al riquadro **Baud Rate**, per un limite massimo di 31250 bps.
Come ovvio, l'impostazione va poi registrata nel file **System - Configuration**, operazione svolta a cura dello stesso Preferences: sarà sufficiente uscire dal programma mediante l'opzione **Save**.
Nel caso si adoperi, per il boot di Amiga, un proprio disco personalizzato, e si intenda adoperare **Ser:**, è indispensabile che nella direcory Devs del disco sia presente il file **Serial.device**, normalmente presente nell'omonima directory del disco Workbench.
Quanto all'uso in scrittura del device Ser:, sempre da ambiente Dos, è in tutto e per tutto analogo a Par: (si veda descrizione specifica).

NIL:

Tra quelli fin qui esaminati, è certamente il device più... strano. **Nil:**, infatti, derivazione del latino **Nihil**, letteralmente vuol dire "nulla".
Ed è, in un certo senso, proprio quello che fa.
Tuttavia anch'esso può risultare utile, soprattutto nell'ambito di batch files.
Redirigendo verso questo dispositivo un qualunque comando del Dos, si evita, per esempio, che il suo normale output venga visualizzato nella finestra corrente. In pratica, a fini più che altro estetici. Un tipico esempio di applicazione può essere rilevato andando a leggere la startup-sequence del disco Workbench con...
Type Workbench1.3:s/startup-sequence
Si noterà come sia presente, tra le altre, una istruzione **ff >nil: - 0**, che serve a velocizzare l'output di testo.
La redirezione verso Nil: evita semplicemente che sia stampato a video il messaggio di copyright del programma, che può essere visionato impartendo solo **ff** in una finestra Shell o Cli.
Un esempio di possibile utilizzo di Nil: adoperato in output è riportato nella descrizione del comando **Which**.
In qualche raro caso, è anche possibile sfruttare il device in **Input**, grazie alla proprietà di inviare, in questo caso, un "End of file".
La cosa può risultare più chiara esaminando un batch file che sfrutta entrambe le modalità di utilizzo di Nil:.
Il file, che chiameremo **XFormat**, una volta mandato in esecuzione tramite **Execute** (o invocandone il solo nome se ne viene settato il bit script), consente di formattare più dischetti in rapida successione, senza digitare continuamente la sintassi del comando Dos.
Si presti attenzione al fatto che, una volta premuto il Return alla prima richiesta, non viene più proposta alcuna conferma, proprio per rendere più veloce il tutto.
Inoltre, il file batch così com'è funziona sul secondo drive (**df1:**). Chi possiede il solo drive interno, dovrà anzitutto modificare "df1:" in "**df0:**", ed inoltre dovrà copiare in **Ram Disk** il comando Format con...
Copy Workbench1.3:system/format Ram:
...e sostituire nel batch il comando **Format** con **Ram:Format**.
Ecco il nuovo comando, come sempre da copiare (con Ed o altri Ascii editor) e salvare su disco:

```
Lab loop
Echo "inserisci disco in df1:"
Ask "e premi Return"
Format >nil: <nil: drive df1: name VUOTO
Ask "FATTO. Ancora? (y/n)"
If warn
Skip loop back
endif
```

Il file formatterà tutti i dischi con lo stesso nome (**Vuoto**), che potrà poi essere modificato in un secondo tempo, se lo si desidera. Come si noterà, si usa il comando **Ask** al solo scopo di attendere la pressione del Return, dopodiché viene attivato Format (vedi anche Amigafacile n. 76), con una doppia redirezione. **>Nil:** evita semplicemente che venga visualizzata la stringa che normalmente segue il caricamento del comando (**Insert.... and press return**).
A questo punto, però, il computer resterebbe ugualmente in attesa di un secondo return, dopo quello da noi richiesto tramite Ask.
Ecco allora lo scopo di **<Nil:**, che provocherà un invio automatico dello stesso, innescando definitivamente la procedura di formattazione.
Ad operazione conclusa, viene poi richiesto se ripetere o meno l'operazione su altri dischi: **"Y"** (e return) provocherà un salto all'inizio del batch, mentre **"N"** concluderà il tutto.

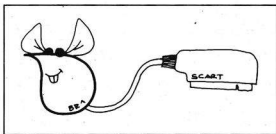
Se i dischi da formattare fossero già in formato Amiga, è consigliabile aggiungere in coda al comando format l'opzione **Quick**, ed eventualmente anche **Noicons** se non si desidera la creazione della directory Trashcan.

RAW:

Più per onor di cronaca, che per altro, va citata l'esistenza di questo device. Non che sia inutile in assoluto: i linguaggi di programmazione possono farne un uso proficuo, ma non altettanto si può dire dell'ambiente Dos.
In pratica, può essere paragonato ai device Con: e Newcon:, e può essere adoperato con le stesse modalità, solo che quanto inviato non viene per nulla filtrato, un po' come per newcon, ma con una importante differenza: se adoperato (p. es.) per redirigere verso un file l'ouput della sua console, non filtrerà neanche il **Ctrl+**, ovvero la fine delle operazioni, con la conseguenza che il file non potrà venir chiuso regolarmente.

Si provino, a titolo puramente dimostrativo, gli esempi proposti per Con: e Newcon:, preparandosi però a dover resettare il computer per riprenderne il controllo.

CON: NEWCON:

Senza rendercene conto, sono questi i device che adoperiamo maggiormente, soprattutto seguendo questa rubrica. **Con:** sta infatti per **Con**sole, mentre **Newcon:** non è altro che una revisione migliorata dello stesso.
In pratica, i device sono deputati a ricevere in input quanto digitiamo da tastiera o imponiamo col mouse, mentre l'output è rappresentato dalla finestra video nella quale il tutto viene visualizzato. Quella che comunemente chiamiamo un finestra **Shell**, non è altro che una console gestita da **Newcon:**, mentre una finestra **Cli** sarà gestita da **Con:**. La differenza tra i due device, come immediatamente verificabile, consiste soprattutto nelle diverse capacità di editing nella finestra, che in fin dei conti si traduce nella più ampia accettazione di caratteri da tastiera nel caso di Newcon: (i tasti cursore semplici o shiftati, la visualizzazione dell'escape, eccetera).
Va però precisato che, mentre il device Con: (e quindi il Cli) è disponibile direttamente da sistema, Newcon: opera modificando il già esistente Con:, quindi necessita di essere abilitato mediante il comando **Mount**. Nel caso del disco Workbench, per esempio, tale procedura è inserita nella startup-sequence, che provvede anche a rendere residente il file **Shell - Seg**, operazioni tutte necessarie per disporre di Shell.
Schematizzando, per disporre di Con: non sono necessarie particolari attenzioni, mentre per adoperare Newcon: e Shell, il disco di boot deve obbligatoriamente contenere:
Il file Newcon-handler nella directory L.
Il file Shell-seg, sempre nella directory L
Il file Mountlist nella directory devs.
Questa sequenza di comandi inserita nella startup sequence:
Resident Cli l:Shell-seg system pure
Mount Newcon:
Nel caso qualcosa "andasse storto" nel tentare di aprire una finestra Shell, cioè una console Newcon: (vedi comando **Newshell**), il sistema si limiterebbe comunque a segnalare la cosa, e ad aprire una normale finestra cli, cioè una console Con:.
C'è ancora da aggiungere che, volendo, è possibile creare autonomamente delle console verso le quali, per esempio, redirigere l'output di un comando Dos. Per far ciò, è sufficiente citare il device desiderato esattamente con le stesse modalità richieste dai comandi Newshell e NewCli, anche se la cosa non sempre può risultare utile.
Per esempio, da una finestra shell si provi a digitare un comando così strutturato:
Dir >Con:0/120/640/100/MIA opt a
Si potrà constatare l'apertura di una nuova finestra con titolo "mia", nella quale verrà visualizzato l'output del comando dir.
A meno di non fermare il listing premendo un tasto, questa però scomparirà immediatamente dopo l'esecuzione di Dir, a conferma della scarsa comodità di tale tecnica. Qualcosa di meglio, però la si può ottenere. Ad esempio, si è in altre occasioni descritta la possibilità di adoperare Shell come un mini editor.
Ecco dunque come averne uno sempre a disposizione, mentre da Shell si può continuare a sbrigare altre faccende:
Run Type Newcon:0/120/640/100/Editor to Ram:test
Trattandosi di editing, stavolta si è adoperato Newcon:, ben più versatile di Con: nella stesura di testi.
La nuova console, aperta in multitasking (= lasciando libera la finestra chiamante) può essere ridimensionata e disposta dove più aggrada sullo schermo, ed al suo interno può essere editato un testo che verrà memorizzato nel file di nome **Test**, in Ram Disk (o dove altrimenti specificato, naturalmente).
Per adoperare l'editor, basta clickare al suo interno per attivare la finestra (appena invocata sarà comunque già attiva) e digitare quanto voluto, senza dimenticare che le funzioni di editing saranno quelle della Shell.
A conclusione di ogni riga occorre quindi inserire un Return, mentre la pressione di Ctrl+\ (barra inversa) chiuderà l'editor (mentre la finestra Editor è attiva!), ed il file in Ram sarà bell'e pronto.
Niente di paragonabile ad un word processor, ma questa tecnica può risultare molto utile per stilare qualche batch file, e nel contempo adoperare la finestra Shell per altri scopi collaterali.
Si ricordi, inoltre, che la console della finestra corrente può essere indicata adoperando il simbolo di asterisco (*), così come applicato nella descrizione del device Par:.

PAR:

Tramite questo nome di device (abbreviazione di PARallel), è possibile inviare un output alla porta parallela di Amiga, inizializzata dal sistema in accordo alle specifiche dello standard Centronics, al quale si adegua anche da un punto di vista fisico.
Tale porta, da ambiente Dos, può essere sfruttata solo per un output diretto verso la stampante, mentre non è implementabile alcuna forma di input.
Concretamente, l'output può essere inviato adoperando i comandi Copy e Type, unitamente alle loro opzioni di redirezione. Traducendo in pratica, questo significa che, per esempio, un comando...
COPY Nomefile TO PAR:
...invierà alla porta parallela il contenuto del file Nomefile, provocandone in definitiva la stampa su carta. Il meccanismo operativo risulterebbe identico nel caso di una stampante seriale, ovviamente con Ser: a sostituire Par:.
Con lo stesso meccanismo, è anche possibile inviare direttamente l'output della finestra Shell alla stampante, adoperando un comando Copy * to prt:. In questo caso, si otterrà la stampa di ogni riga digitata alla pressione del Return, mentre Ctrl+\ provocherà un ritorno alle normali attività della Shell.

Lo stesso risultato si otterrebbe adoperando Type al posto di Copy, senza alcuna differenza, o sfruttando il simbolo di redirezione dell'output concesso dal dos (">").
In quest'ultimo caso, il comando prima visto per stampare un file diventerebbe Type >par: Nomefile, senza dimenticare un eventuale percorso (p.es. Ram:Nomefile).
Naturalmente Nomefile deve indicare un file ascii di testo, tuttavia è possibile inserire in esso dei codici di controllo che modifichino il comportamento della stampante.
Tali codici, si badi bene, devono essere eventualmente inseriti direttamente nel testo adoperando degli editor che consentano di immettere anche codici ascii non corrispondenti ad alcun carattere alfanumerico (si veda postamiga del numero scorso).
Se, per esempio, si volesse ottenere una stampa in formato espanso, occorrerebbe inviare un codice 14 alla stampante. Ciò comporterebbe in pratica il dover immettere nel file di testo un vero Chr$(14).
Adoperando la porta parallela, inoltre, possono eventualmente essere inviati alla stampante delle sequenze di Escape per ottenere particolari effetti grafici, ma attenzione (!!), queste non corrisponderanno al codice Ansi di Amiga, ma ad eventuali particolari codici implementati dalla stampante stessa (se esistenti, descritti nel manuale della stampante).

Per concludere, non si dimentichi che anche la gestione di Par: da parte del sistema passa attraverso l'utilizzo di un handler che risiede sul disco Workbench: il file Parallel.device, memorizzato nella sua directory Devs.

PRT:

Sebbene il device **Par:** (o **Ser:** per modelli seriali) consenta di adoperare da ambiente Dos una stampante, tuttavia non sempre può risultare comodo. Come deducibile dalla descrizione, qualunque particolare settaggio o implementazione di caratteristiche particolari, vanno inviati esplicitamente assieme al testo da stampare.
AmigaDos, però, permette di sfruttare un altro device molto più utile, **Prt:**, espressamente dedicato alla gestione della stampante.
Intanto, indipendentemente dal fatto che essa sia collegata alla porta seriale o a quella parallela, l'output verrà in ogni caso diretto verso di essa; tutte le specifiche che la riguardano, infatti, Prt: le preleva dal file **System - Configuration**, notoriamente modificabile dal programma Preferences.
Ciò si traduce nella totale indipendenza del testo da inviare in stampa, che non necessita di codici specifici per un certo tipo di stampante. Sempre tramite Preferences, viene infatti precisato il modello adoperato, ed eventuali altri particolari settaggi (numero di righe, fogli singoli o a modulo continuo, marginatura, eccetera).
In pratica, mentre con Par: e Ser: l'output viene inviato così com'è, con Prt: viene invece "filtrato" ed "adattato" alla stampante in uso. Ciò consente, tra l'altro, di adoperare lo standard **Ansi** per ottenere la sottolineatura, il grassetto, ed altro, inserendo magari i codici di **Esc**ape nel testo, sicuri della loro "trasportabilità".
Anche per l'uso di Prt: è indispensabile la presenza nel device logico **Devs:** (corrispondente in genere alla directory **Devs** del disco di boot) di un file manipolatore, **Printer.device**, che a sua volta presuppone la presenza del "driver" (un file specifico) legato al particolare modello di stampante nella subdirectory **Printers**.
C'è da aggiungere che, normalmente, il device Prt: modifica il carattere usato da Amiga per indicare un "a capo" (il cosiddetto **Linefeed**, ovvero il codice **Ascii 10**) trasformandolo nella sequenza Return + Linefeed (un codice ascii 13 seguito da 10). In talune stampanti, evenienza in verità abbastanza rara, può risultare utile evitare questa performance. In questo caso basterà usare la forma **Prt:Raw** nel definire il device, ed il Return non verrà aggiunto al linefeed.

Da un punto di vista formale, l'uso del device è analogo a quanto descritto a proposito di Par:.

NEWSHELL E NEWCLI

Questi due comandi svolgono in pratica la stessa mansione, cioè quella di aprire una finestra Dos verso la quale dirigere l'input di tastiera. **Newcli** renderà però disponibile una finestra Cli, ovvero una console riferita al device **Con:** (si vedano descrizioni specifiche), mentre **Newshell** farà riferimento a **Newcon:**, con tutti i vantaggi che ne derivano: possibilità di adoperare i tasti cursore per spostamenti laterali non distruttivi, richiamo righe di comando immesse in precedenza (tasto cursore in alto), capacità di visualizzare (e quindi editare) il codice di **Esc**ape, eccetera.

Ogni finestra aperta tramite questi comandi è dotata di un task proprio, il che si traduce in una completa indipendenza da eventuali altre finestre. Il numero di task, salvo manipolazioni ad opera dell'omonimo comando, viene indicato dal prompt della finestra. Volendo inserire una versione personalizzata del prompt, è possibile riferirsi al numero di task precisando i caratteri **%N**, validi tanto per il Cli che per la Shell.
Quest'ultima, in più, consente l'indicazione della directory corrente nel prompt, attivata dalla presenza dei caratteri **%S** nella definizione del Prompt (si veda descrizione sul manuale).
Una finestra Dos, quale che sia la sua origine (cli o shell), va richiusa tramite un comando **Endcli**.

NEWSHELL NEWCON:x/y/largh/alt/titolo FROM file

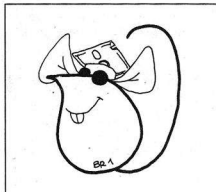
From

Adoperando la keyword From (obbligatoria se si adopera l'opzione **File**) seguita dal nome di un file batch (comprensivo di eventuale percorso!), questo verrà immediatamente eseguito appena aperta la finestra dos, al posto del già citato Shell-Startup, prima di passare il controllo all'utente.
La cosa può avere risvolti molto interessanti, soprattutto se si considera la totale indipendenza tra una finestra e l'altra. Da un punto di vista pratico, si può per esempio preparare una serie di files che svolgono compiti differenti, quindi lanciare **Newshell** (o **Newcli**), in accordo alle esigenze del momentro, attivandone con From uno piuttosto che l'altro, o adoperandoli tutti, ma in più finestre.
Vediamolo in pratica, risulterà molto più chiaro.
Anzitutto, si preparino due Script File in **Ram Disk** (o su un dischetto, se lo si preferisce) sfruttando la tecnica esaminata nella descrizione di **Con:**, considerata la loro brevità. In altre parole, si impartisca...
*Type * ram:file1*
...e return. A questo punto, si copino le due righe seguenti, immettendo un return alla fine di ogni riga e **Ctrl+** per concludere la scrittura:
Echo "Io sono il file1"
*Prompt "%N) *e[37;41;1m%S*e[0m->"*
Ora, dopo aver premuto Ctrl+\, si ripeta l'operazione per editare un file di nome File2 *(Type * to Ram:File2)*, così configurato:

Echo "Io sono il File2"
*prompt "%N) *e[32;41m%S*e[0m->"*

Dopo l'ennesimo Ctrl+\, si digiti nella finestra shell il comando...
Newshell From Ram:file1
...che provocherà l'apertura di una nuova finestra, caratterizzata da un prompt colorato ed in reverse. La si sposti un po' sullo schermo, quindi si digiti al suo interno...
Newshell From Ram:file2
Apparirà un'altra finestra shell, anch'essa con il prompt in reverse, ma di diverso colore.
Quanto visto a titolo di esempio, si consideri che può essere allargato a condizioni ben più importanti di un semplice fatto estetico legato al prompt. Per esempio, si potrebbe stilare un batch che copi in Ram tutti i comandi del dos, e riassegni il device C: alla Ram:. Aprendo una finestra che utilizza nel parametro From un file di questo genere, si avrebbe (p. es.) quella particolare finestra che adopera i comandi in Ram (quindi con maggiore velocità), mentre un'altra potrebbe sfruttare il device C: di un disco inserito nel secondo drive.
Le applicazioni possibili, insomma, si fanno davvero tante.

Newcli - Newshell

Il comando (**Newcli** oppure **Newshell**) può essere impartito senza alcun parametro. In questo caso verrà aperta una finestra dalle dimensioni fisse, grosso modo al centro dello schermo, e verrà eseguito, se presente nella directory **S** del disco di boot (ovvero il device logico S:), un eventuale file batch di nome **Shell-Startup** (o Cli-startup). Nel disco Workbench il file provvede ad impostare il prompt e ad inizializzare una serie di **Alias** (se ne parlerà in un prossimo appuntamento). Per dargli un'occhiata, basterà impartire...
Type Workbench1.3:s/shell-startup
Adoperando Newshell (o Newcli) senza altre specifiche, è ovvio che tutte le nuove finestre aperte importeranno le stesse caratteristiche, in accordo appunto con il file di cui sopra.

NEWCLI CON:x/y/largh/alt/titolo FROM file

x/y/largh/alt/titolo

Con questo parametro è possibile impostare le **dimensioni** della finestra che verrà aperta, ed un suo eventuale **titolo**.
A seconda che si desideri accedere al **Cli** oppure a **Shell**, va inoltre precisato il relativo **device**, che nel caso di **Newcon:** deve essere stato preventivamente installato con **Mount** (si veda ampia descrizione nelle pagine specifiche).
A seguire, **X** ed **Y** identificano la coordinata orizzontale e verticale dell'angolo superiore sinistro della finestra, **Largh** la larghezza ed **Alt** l'altezza.
Il tutto espresso in pixel, e badando di non superare i limiti massimi consentiti dallo schermo.
Ognuno di questi valori può, volendo, essere omesso (verrà assunto un valore **0**), nel qual caso è comunque necessario non tralasciare la corrispondente barra obliqua. Anche il titolo è facoltativo, ed anche in questo caso risulta obbligatoria la presenza della barra obliqua che lo precede. Se poi questo contiene al suo interno qualche spazio, è indispensabile racchiudere TUTTA la definizione tra doppi apici.
Per esempio...
Newshell Newcon:0/0/640/250/Work
...aprirà una finestra grande quanto l'intero schermo (considerando la risoluzione ad **80** colonne) con titolo **Work**. Stesso risultato, però, si otterrebbe anche con...
Newshell Newcon://640/250/Work
Se, poi, invece che Work volessimo adoperare come titolo **My Work** (con uno spazio intermedio), si renderebbe necessaria una forma...
Newshell "Newcon://640/250/My Work"
Per escludere il titolo, la stessa sintassi andrebbe invece così digitata:
Newshell Newcon://640/250/

L'eccessiva verbosità dei parametri porta in genere l'utente ad ignorarne i vantaggi, preferendo adoperare un semplice **Newshell**, per poi ridimensionare la finestra con il mouse.
Grazie alla versatilità del nuovo device (newcon), è possibile assegnare l'intera sintassi ad un comando **Alias**, ovvero impartire in modo diretto o (ancora meglio) inserire nella startup-sequence (oppure nell'eventuale Shell-startup) un comando come...
Alias Shell Newshell Newcon://640/250/
...per avere sempre una finestra a tutto schermo, ogni qualvolta si impartisce Shell in sostituzione di Newshell.
Decisamente più comodo.

di Luigi Callegari

I BASIC COMPILATI

Chi è abituato al Basic conosce i suoi limiti, soprattutto per ciò che riguarda la velocità di elaborazione; un buon compilatore riesce

Sin dai tempi del lancio dell'Amiga 1000 con Kickstart e Workbench V1.1, il nostro gioiello a 16 bit è sempre stato fedelmente accompagnato da un interprete Basic. Ai primordi fu effettivamente ABasiC, un dialetto piuttosto rozzo, svilppato in tempo ridottissimo dalla inglese Metacomco giusto per il "lancio" negli USA.

Si trattava di un Basic minimo, piuttosto veloce, specie se raffrontato con i Basic che si erano sino ad allora visti su macchine ad otto bit, ma senza alcun supporto diretto delle caratteristiche hardware di Amiga.

In Italia Amiga fu presentato fin dall'inizio corredato di **AmigaBasic**, interprete sviluppato dalla **Microsoft**, che adattò la versione approntata per la linea professionale Apple Macintosh che, a sua volta, aveva molte somiglianze con il **QuickBasic** dell'ambiente **Ms-Dos**.

Tra pregi e difetti di AmigaBasic, noti a chiunque lo abbia usato, si è giunti all'attuale scelta di mercato rappresentata dall'Amiga 3000 che viene fornito **senza** alcun linguaggio Basic. I motivi della scelta sono molteplici, dichiarati e non dichiarati da mamma Commodore: è quindi preferibile rivolgersi direttamente al mercato per trovare un Basic adatto ai propri scopi (tra cui ricordiamo **Amos, GFA, True Basic, HiSoft Basic, F-Basic, Blitz Basic...**) tutti più o meno superiori al vetusto AmigaBasic.

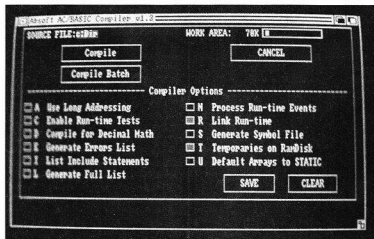
Una macchina professionale quale dovrebbe esser l'Amiga 3000, infatti, non può essere fornita con un linguaggio poco efficiente come AmigaBasic, ma viene invece fornito con **ARexx**, in grado di sostituirlo benissimo per applicazioni minime; del resto AmigaBasic è **incompatibile** con i processori **68030** usati da schede velocizzatrici e da Amiga 3000.

Sono in circolazione vari **compilatori Basic**, alcuni dei quali supportano direttamente AmigaBasic nel senso che, dando loro in pasto un listato battuto con l'interprete AmigaBasic, generano (senza "lamentarsi" troppo) un codice oggetto, rappresentato da istruzioni in linguaggio macchina 68000, eseguibili direttamente dal processore alla massima velocità possibile.

In queste pagine riportiamo alcuni cenni per un rapido utilizzo di alcuni compilatori per Amiga tra i più diffusi: AC/Basic Compiler (V1.3), HiSoft Basic Compiler (V1.05), GFA Basic Compiler (V3.51e) e F-Basic (V2.0).

Si noti che GFA e F-Basic, pur essendo di gran lunga i più efficienti dei quattro, sono tuttavia del tutto **incompatibili con AmigaBasic**. Infatti è necessario sviluppare i "sorgenti" usando sintassi differenti da quelle previste dall'interprete originale Commodore.

AC / Basic Compiler

E' stato storicamente il primo compilatore Basic per Amiga disponibile commercialmente, aggiornato solo tre volte dalla **Absoft**, la softhouse che lo ha prodotto e che è famosa più per avere dato ad Amiga **AC/Fortran** (l'unico compilatore Fortran di sufficiente qualità di-

Installare compilatori

I compilatori per Amiga vengono forniti, solitamente, su dischetti di tipo "bootable", ovvero inseribili direttamente all'accensione od al reset del computer per inizializzarlo. Cio' comporta la presenza, sui dischetti, di una grande quantita' di dati di supporto, come i sorgenti di dimostrativi, che non sono comunque strettamente necessari per fare funzionare il compilatore. A beneficio di chi volesse costruirsi propri dischetti di lavoro, oppure realizzare sul proprio Hard Disk directory contenenti i soli files necessari per lavorare, riportiamo, per ognuno dei pacchetti presentati, l'elenco completo(?). Si noti che, nelle differenti versioni, le lunghezze dei files possono cambiare e che l'elenco riportato consente di avere un ambiente di lavoro "minimo" per compilare programmi, ed eseguirli, senza citare eventuali programmi di supporto evoluto.

AC/Basic Compiler (V1.3)		
AC-Basic	17188	Compilatore
bas.dl	43264	(L) Libreria
bas.rl	43264	(L) Libreria
basini.sc	428	(L) Supporto libreria
bas001.bc	29440	(L) Libreria
bas002.bc	35584	(L) Libreria
bas004.bc	13312	(L) Libreria
loader.sc	1024	(L) Supporto libreria
HiSoft BASIC Compiler (V1.05)		
HiSoft BASIC	24636	Controllo compilatore
HB.Compiler	107168	Compilatore Basic
hisoftbasic.library	48908	(LIBS) Libreria
arp.library	17100	(LIBS) Libreria
GFA Basic Compiler (V3.51E)		
GFABasic	135956	Interprete Basic
GFA_Bcom	89572	Compilatore Shell
Gl	6920	Linker
GfaLib	145112	Libreria compilatore
GfaLibrary	68802	Libreria linker
GfaLibrary.index	126456	Libreria linker
MenuX	34060	Controllo compilatore
MenuX.info	386	Icona controllo compilatore
MenuX.GFA	9636	Sorgente controllo compilatore
Delphi Noetic F-Basic (V2.0)		
FB	167764	Compilatore Shell
Link	10984	Linker Shell
FastError	8385	Libreria errori
FastIFF	6968	Libreria IFF
FastLib	39812	Libreria standard
FastLinkLib	41148	Libreria linking
FastSysLib	33680	Libreria sistema
SLDB	50412	Debugger

I files dei compilatori Basic

I vari compilatori, per funzionare correttamente, necessitano di alcuni files che devono, quindi, esser presenti sul dischetto. Nella tabella vengono riportati i files più importanti, la loro lunghezzae spressa in bytes e la funzione che svolgono.

sponibile per Amiga) che per le prestazioni del compilatore di cui ci occupiamo.

AC/Basic è, infatti, piuttosto *rozzo*, genera un codice ricco di "overhead", ovvero dotato di un elevato numero di bytes necessari per supportarne l'esecuzione (**runtime**). E' compatibile al 99% con AmigaBasic, ma non prevede estensioni particolari allo standard del linguaggio.

Il suo utilizzo è molto semplice, anche da Workbench. Si clicca due volte sulla sua icona per fare comparire lo schermo di lavoro. Da menu si sceglie **Open** per fare comparire il requester di file "minimo", tristemente simile a quello di AmigaBasic e ben lontano, per comodità, da quello, ad esempio, della ARP library usato da HiSoft Basic.

Si digita il nome del file sorgente, ovvero il nome del file che redatto a parte con un normale editor ASCII (MicroEmacs, Ed, TxEd, CygnusEd, Az, Fed...) e salvato preferibilmente su disco. Dopo l'immissione del nome, appare un secondo requester che consente di regolare le opzioni di compilazione. Con il mouse si può commutare un gadget booleano (acceso / spento) accanto ad una serie di opzioni identificate da una lettera dell'alfabeto (A...U) e dalla spiegazione esplicita.

A: Use Long Addressing

Utilizza, nel codice eseguibile, l'indirizzamento completo a 32 bit. Ciò consente di non avere alcuna restrizione virtuale sulle dimensioni del programma e dei dati, ma genera un codice eseguibile più lungo e più lento in funzione del tipo di programma. In genere, per programmi brevi, è preferibile lasciarla disattiva.

C: Enable runtime tests

Inserisce, nel codice eseguibile, istruzioni supplementari che verificano e segnalano eventuali errori durante l'esecuzione del programma. Di regola, infatti, il codice eseguibile non si preoccupa di verificare se, ad esempio, l'indice di una matrice esista veramente in memoria oppure che non sia esaurito lo spazio sullo stack dopo numerosi annidamenti di subroutines. Con questa opzione si evitano alcuni **Guru**, in taluni programmi non protetti completamente contro gli

errori, ma si ottiene un codice più lungo e più lento in esecuzione.

D: Compile for Decimal Math

Per default, quando questa opzione è disabilitata, il compilatore genera un codice per le operazioni matematiche in virgola mobile utilizzanti il formato IEEE. Questo tipo di formato è più veloce in esecuzione, ma meno preciso. Attivando l'opzione il programma compilato userà una notazione interna decimale, più dispendiosa da gestire, ma più precisa.

E: Generate Error List

Il compilatore scrive in un apposito file la lista degli errori incontrati durante la compilazione, altrimenti soltanto mostrata a video in "tempo reale" durante la compilazione.

I: List Include statements

Per default il compilatore non lista eventuali files ASCII inclusi nel sorgente. Abilitando l'opzione, invece, le istruzioni di inclusione verranno sostituite nel listato dal corpo del file incluso.

L: Generate full list

Per default, il compilatore genera il listato in modo compatto; abilitando l'opzione si ottiene un listato più "verboso".

N: Process runtime events

Alcune istruzioni di AmigaBasic (**On Mouse**, ad esempio) richiedono questa opzione che abilita l'interruzione dell'esecuzione del programma principale per trattare una routine di servizio dell'interruzione.

R: Link runtime

Per default il compilatore genera un file eseguibile piuttosto compatto, ma che abbisogna dei files di libreria di **AC/Basic** nell'apposita directory **L:** (vedi riquadro) per funzionare. Abilitando l'opzione si genera, invece, un programma di tipo "stand alone", ovvero funzionante senza bisogno di altri files di supporto.

Si ottiene, però, un file più lungo. Senza questa opzione è possibile inserire più files compilati su di uno stesso disco inserendo, in **L:**, una sola libreria di runtime valida per tutti, con evidente risparmio di spazio.

S: Generate Symbol File

Abilitando questa opzione il compilatore genera un file contenente tutti i simboli usati nel listato (nomi di variabili, etichette...).

T: Temporaries on ram disk

Per default il compilatore scrive i propri files di lavoro (creati e cancellati in modo trasparente all'utente) nella stessa directory del file sorgente.

Abilitando questa opzione, invece, si ottiene la scrittura automatica di questi files nel disco RAM, in modo da aumentare al massimo la velocità di compilazione.

L'unico motivo per disattivare l'opzione è dovuto alla disponibilità di una quantità di Ram insufficiente per conservare, oltre al sorgente ed al compilato, anche i files temporanei del compilatore.

U: Default arrays to STATIC

Per default, i vettori e le matrici definite nel listato sorgente sono assunte come dinamiche. Abilitando questa opzione vengono interpretate per default (salvo esplicita dichiarazione contraria nel sorgente, ovviamente) di tipo statico.

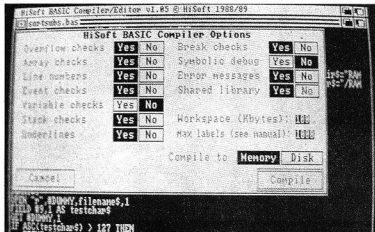
Una volta indicate le opzioni di compilazione necessarie, si può cliccare sul gadget **Save** per salvare (sempre nella directory **L:**) in un file chiamato **Ac-basic.opt** le preferenze che verranno usate, per default, in tutti i successivi caricamenti di AC-Basic.

Cliccando, invece, su **Clear** si azzerano le opzioni impostate, mentre con **Cancel** si ritorna allo schermo vuoto precedente col menu di Open. Per compilare si clicca sul gadget **Compile**. In caso di errori, si deve rientrare nell'editor ASCII, eseguire le correzioni, rientrare in AC- Basic, ricaricare il listato e ripetere le operazioni.

HiSoft Basic Compiler

Il compilatore prodotto dalla **HiSoft** è completamente **compatibile** con lo standard AmigaBasic ma prevede alcune estensioni. Ciò significa che, teoricamente, qualunque programma che funziona senza problemi *sotto* l'interprete AmigaBasic, può essere compilato tranquillamente da HiSoft Basic. Invece un programma sviluppato con HiSoft Basic può non funzionare sotto interprete, soprattutto se si usano le specifiche esclusive del compilatore, riportate nel manuale HiSoft Basic.

Il compilatore, realizzato in diverse versioni, può essere invocato da Shell (HB.Compiler) o da Workbench con un doppio click sull'icona apposita. Da Shell si deve impartire una sintassi di compilazione piuttosto farraginosa, simile a quella dei compilatori **C**. Da Workbench risulta tutto più semplice ed intuitivo.

Lanciato il programma da Workbench, ci si ritrova nell'editor **HiSoft**, uguale a quello di tutti i prodotti di questa casa per Amiga (come, ad esempio, l'assemblatore **DevPac**). A questo punto si può

digitare il listato sorgente in Basic usando le funzioni di redazione presenti nei vari menu per correggere, indentare, spostarsi, eccetera. Oppure si può scegliere l'opzione **Load** del primo menu per fare comparire il requester di file della **ARP Library** (della quale avremo ricopiato il file nella directory **LIBS**, vedi riquadro sull'installazione). Qui si può scegliere il nome di un file ASCII (magari scritto con l'interprete AmigaBasic e salvato con l'opzione **Save "nomefile", A**) da caricare con l'editor.

Per compilare si può scegliere **Compile** da menu, oppure premere la combinazione di tasti **Amiga** e **C**. Appare così il requester di opzioni di compilazione, simile a quello di AC/Basic.

Vediamo, in breve, il significato delle sue opzioni, rammentando che sono tutti gadget booleani (sì, no), che cioè abilitano o disabilitano semplicemente l'opzione. Le spiegazioni sono riportate in forma stringata dal momento che sono del tutto simili a quelle già viste in precedenza.

Overflow checks
Verifica di overflow (eccedenza di capacità) numerico.

Array checks
Verifica dei riferimenti agli elementi di vettori e matrici.

Line Numbers
Numerazione delle linee del sorgente.

Event checks
Verifica e servizio di eventi asincroni (mouse, menu...) durante l'esecuzione.

Variable checks
Controllo dei riferimenti a variabili predefinite.

Stack checks
Controllo dello spazio disponibile sullo stack in esecuzione.

Break checks
Controllo della pressione della combinazione di tasti per il break (**Ctrl-C**).

Symbolic Debug
Generazione informazioni supplementari per il debugging in apposito file.

Error Messages
Messaggi alfanumerici per segnalare errori in esecuzione.

Shared Library
Genera un file eseguibile solo con il file di libreria supplementare sul disco oppure di tipo "stand alone".

Workspace
Qui si specifica il numero di kilobytes assegnati come area di lavoro del compilatore. Per default vale 100K, ma è modificabile se non si dispone di molta RAM o si compilano programmi lunghi.

Max Labels
Si specifica il massimo numero di etichette alfanumeriche di riferimento utilizzabili nel programma. Per default vale 1000, valore sufficiente per la maggior parte delle applicazioni; non richiede una eccessiva quantità di memoria.

Infine, si deve indicare se si desidera compilare in memoria Ram o su disco.

Nel primo caso il file oggetto viene scritto solo in RAM e può essere eseguito con il comando **Run** (oppure **Amiga + X**) direttamente dall'editor di HiSoft Basic.

Nel secondo caso il modulo eseguibile verrà scritto su disco nella stessa directory dalla quale è stato letto il file sorgente (ma non sarà eseguibile dall'editor con il comando Run, ma solo dallo Shell o dal Workbench).

GFA Basic Compiler

Il compilatore della **GFA Systemtechnick** presenta numerosi vantaggi rispetto ad HiSoft ed AmigaBasic. E' estremamente veloce e dispone di istruzioni molto sofisticate. Il suo compilatore richiede un listato sorgente scritto appositamente in GFA Basic (con sintassi diversa da Amigabasic) e salvato su disco con l'interprete non in formato ASCII.

Il compilatore può essere invocato direttamente da Shell/CLI oppure tramite il programma di supporto, scritto in GFA, chiamato **MenuX**, che fornisce un controllo completo dell'ambiente integrato "compilatore ed interprete" direttamente da Workbench.

Ci riferiremo al funzionamento del compilatore da Shell, in quanto il programma MenuX (del quale si può trovare il sorgente sul disco stesso del linguaggio) usa, in modo **intuition**izzato, le stesse opzioni, selezionabili tramite appositi gadget. Per compilare un listato sorgente scritto con l'interprete GFA e salvato in formato interno (non ASCII) si usa una linea del tipo...

GFA_Bcom Monica S&

...che compila il file sorgente chiamato **Monica** usando l'opzione **S&** (di cui ci occuperemo tra breve), generando per default il file oggetto intermedio **Test.o.** Si noti che, a differenza di altri compilatori, il nome del file intermedio non è assegnato in base a quello del sorgente, ma vale **sempre** "Test.o" se non si usa l'apposita opzione di ridenominazione esplicita. Vediamo in breve le opzioni di GFA.

%0 - Esegue le divisioni intere come tali se dopo il risultato viene usato come intero.

%3 - Esegue sempre le divisioni intere come tali.

%6 - Esegue addizioni intere miste come somme in virgola mobile.

mx - Il programma usa soltanto **X** bytes di memoria.

***&** - Usa **Muls** per moltiplicazioni di longword.

***%** - Non usa Muls per moltiplicazioni di longword.

F% - Il valore di rientro da funzione è sempre un intero.

Xn - Indica che **X** è il nome di una routine da leggere in un modulo di linking.

U - Controlla la pressione di Ctrl-Shft-Amiga una volta sola.

U+ - Controlla la pressione di Ctrl-Shft-Amiga prima e dopo ogni istruzione.

I+ - Abilita routines di interruzione.

I- - Disabilita routines di interruzione.

S& - Parametri di Select e Case a due bytes.

S% - Parametri di Select e Case a quattro bytes.

S> - Ottimizza Select e Case per tempo di esecuzione.

S< - Ottimizza Select e Case per dimensione.

E$ - Usa testi per i messaggi di errore.

E# - Usa numeri per segnalare errori.

P> - Le subroutines devono essere sottoprogrammi GFA.

P< - Le subroutines devono essere compilate per 68000.

F> - Genera specifica Endfunc.

F< - Non genera specifica Endfunc.

C+ - Salva registri A3 - A6 sullo stack

C- - Non salva registri A3 - A6 sullo stack.

N+ - Abilita controllo dello stack.

N- - Disabilita controllo dello stack.

Per trasformare il codice oggetto generato da **GFA_Bcom** (il già citato "Test.O"), in un file eseguibile da Shell o da Workbench, si deve usare il **linker** GL. Ad esempio...

GL -s

...linka automaticamente il file Test.O con il file di startup e la libreria standard (GFA.Library), usando l'opzione di linking **S**, per generare il file eseguibile finale Test. Le possibili opzioni passabili al linker GL da Shell sono poche:

-S - Aggiunge la tabella simbolica al file.

+x - Usa la libreria chiamata "x" al posto di GFA.Library.

Xx - Inserisce il file oggetto "Xx.O" durante il linking.

- Non include il file Test.O nel linking.

-Ox - Linka "x.O" invece dello standard Test.O.

-Px - Genera il programma eseguibile "x" invece di "Test".

-W - Abilita il flag di Wait.

❑❐

F-Basic V2.0

Anche F-Basic usa una sintassi differente da quella di AmigaBasic. I programmi, quindi, devono essere sviluppati appositamente per il suo compilatore e non è possibile compilare listati studiati per l'interprete standard.

F-Basic è solo compilatore, cioè non incorpora nè un editor nè un interprete. L'utente deve sviluppare i suoi programmi con un editor ASCII standard (MicroEmacs, TxEd, Ed, Az, Cygnus...), registrarlo su disco (o in RAM) e poi compilarlo.

I programmi richiedono sempre la presenza del file di libreria per funzionare e tale file non è di pubblico dominio, ma bisogna pagare10 dollari alla **Delphi Noetic** per ogni programma (non per ogni copia!) che si desidera distribuire liberamente con tale file sul disco. Oppure si può linkarli con la libreria per generare un file "stand alone", ma anche in questo caso bisogna richiedere l'autorizzazione alla softhouse pagando una piccola tassa.

Per compilare un file chiamato, ad esempio, **Ormellese** con l'opzione **P**, si usa...

FB Ormellese opt -p

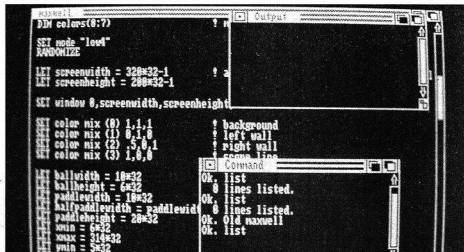
...che genera il file intermedio **Ormellese.bin**. Ovvero, il nome del file intermedio è quello del file di input più il suffisso "**.bin**". Le opzioni previste dal compilatore FastCom (FB) sono pochissime.

-C x - Le stringhe possono essere lunghe al massimo "x" bytes. Per default "x" vale 1000 caratteri.

-G - Esegue il programma, subito dopo la compilazione, senza nemmeno salvarlo su disco.

-O x - Il file oggetto finale può essere lungo al massimo "x" bytes. Per default, "x" vale 50000 bytes.

-P - Scrive sullo schermo tutte le linee durante la compilazione.

-S x - Il buffer di lavoro delle stringhe diventa di "x" bytes. Per default x vale 1000 bytes e tale valore deve essere alterato solo se si incappa in qualche errore specifico durante la compilazione.

di Luigi Callegari

ANIMARE LA GRAFICA

impariamo a realizzare i nostri cartoni animati con il favoloso programma Deluxe Paint III; naturalmente per Amiga

Quando si dice **Deluxe Paint**, tutti pensano subito ad uno stupendo ed originale programma per disegnare pagine grafiche, magari coloratissime ma rigorosamente immobili sullo schermo.

DP3, invece, consente animazioni di disegni creati con un metodo molto vicino al **page flipping** che tutti i buoni programmatori, in particolare quelli che provengono dal **C/64**, conoscono già.

Il fatto è che non molti sanno come realizzare, in pratica, tali animazioni, un po' per la concisione del manuale, un po' per la convinzione (errata) che sia difficile.

Illustreremo in queste pagine, passo per passo, come costruire una breve sequenza animata. Speriamo così di evidenziare come DP3 sia un programma, tutto sommato, immediato ed intuitivo da usare anche per le animazioni e di invogliarvi a sfruttare la caratteristica esclusiva del più diffuso pacchetto per Amiga.

Il progetto

Come quando si scrive un programma, è fortemente consigliabile, per realizzare un'animazione, *pianificare a priori* il da farsi. Così facendo si ridurranno i tempi di lavorazione al computer per ottenere i migliori risultati. Supporremo di realizzare un "cartoncino animato" relativo ad un omino che cammina, immaginando di far ammirare in capolavoro finale in un monitor posizionato nella vetrina di un negozio: sarà quindi necessario che risulti ben dimensionato.

Una scorciatoia tipica consiste nel realizzare il disegno in scala ridotta e poi procedere al suo **ingrandimento** con le apposite opzioni di **brush**.

Dovremo quindi disegnare alcuni **frames** la cui traduzione è: fotogramma. Il nome deriva dalla procedura con la quale vengono effettuate le animazioni, da sempre, anche nel cinema. Una pellicola cinematografica, infatti, è formata da una sequenza di immagini, chiamati appunti fotogrammi, ciascuno dei quali riprende un istante dell'animazione. Essa viene materializzata alla nostra attenzione riproponendoli, ad una certa velocità, sfruttando il ben noto meccanismo di persistenza di un'immagine sulla retina dell'occhio umano.

Scomponendo e semplificando l'immagine dell'omino che cammina, possiamo supporre che bastino **sette** frames, che verranno disegnati singolarmente per esser poi riprodotti, in sequenza, ad una velocità regolabile agendo dall'interno di DP3. Con un pizzico di furbizia, possiamo anche immaginare che due fotogrammi dell'animazione siano identici: dal momento che l'uomo avanza muovendo le due gambe (e le due braccia), il fotogramma centrale di ambedue le fasi dell'animazione (movimento in avanti di gamba sinistra - braccio destro prima e poi movimento gamba destra - braccio sinistro) sarà eguale: l'omino con braccia e gambe allineati verticalmente. Così possiamo evitare il disegno di un fotogramma.

Disegnare

La fase più critica è quella del disegno vero e proprio. Non tutti hanno suffi-

l'idea dello scorrimento. Abbiamo pertanto introdotto alcuni puntini chiari sullo sfondo rettangolare scuro che, durante lo scorrimento, si "spostassero" in senso contrario alla direzione dell'omino dando l'idea del movimento.

Subito dopo ci accorgemmo che l'omino si muoveva, ma, nonostante lo sfondo scorrevole, occorreva inserire un effetto di accentuazione del movimento, tecnica ben nota a chi realizza cartoni animati, si pensi allo strabuzzare degli occhi fuori dalle orbite per l'attenzione, od alle "sgommate" dei personaggi che iniziano a correre. Così abbiamo ridisegnato i fotogrammi (in piccolo) facendo andare su e giù, accorciando ed allungando il blocco del busto del corpo e la testa del nostro personaggio.

ciente abilità per realizzare animazioni ma, purtroppo, questo non possiamo spiegarvela, bisogna averla!

Come si vede nelle foto riportate, l'omino è chiaramente disegnato prima in dimensioni molto piccole, circa **16 pixel** di altezza. Ciascun fotogramma può essere inizialmente disegnato e salvato come **normale grafico IFF** di Deluxe Paint. Per ingrandirlo, si sceglie dal menu laterale **la forbicina**, si ritaglia con il mouse (pressione del pulsante sinistro) la sagoma dell'omino e si sceglie dal menu **Brush** la sub-opzione **Resize/Double**. Ripetendo più volte la funzione si ottiene una sagoma di dimensioni ragguardevoli, ben visibile anche se "seghettata", come si vede dalle foto.

La produzione dei disegni avviene, di solito, per affinamenti successivi. Dopo la prima serie di disegni, ad esempio, ci siamo accorti che, per migliorare l'effetto di spostamento, era necessario inserire uno sfondo (pavimento) che desse

Programmazione

Innanzitutto, per usare le funzioni di animazione di Deluxe Paint III, è indispensabile avere a disposizione della **Fast Ram**. Ciascun fotogramma deve essere presente in memoria durante l'animazione, ovviamente, e ciò richiede una certa quantità di memoria anche per brevi e semplici animazioni, come la nostra.

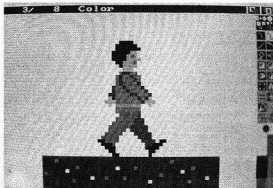
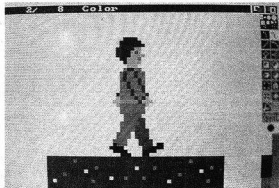
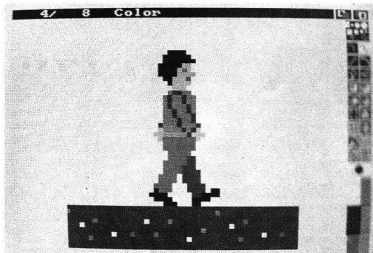
Per creare un **Frame**, si opziona **Anim/Frames / Add Frame**. In questo modo si indica che vogliamo aggiungere un fotogramma all'animazione. Per default, in effetti, lo schermo "attuale" è considerato come primo frame.

Quindi è più corretto pensare: "un fotogramma è già quello sul video, per iniziare a disegnarne un altro devo usare Add Frame".

Occorre poi dichiarare il numero di fotogrammi voluti anche se, in seguito, possiamo modificarne il valore per allungare l'animazione; fissarlo all'inizio, comunque, consente una migliore sistematicità di lavoro ed un più agevole spostamento tra i fotogrammi in memoria grazie alle apposite funzioni di menu. L'ozpione **Set Frame #** fa comparire un requester in cui scrivere il numero di fotogrammi, nel nostro caso **7**.

I vari frames possono essere inseriti disegnandoli al momento ed intercalandoli ad **Add Frame**, oppure caricandoli uno dopo l'altro da disco con la consueta opzione **Picture/Load**, ovviamente sempre intercalata da Add Frame. Sulla barra del menu una coppia di numeri nel formato **X/Y** indica costantemente che è visualizzato il frame numero **X** in un numero globale **Y** (ancora 7, nel nostro caso).

Se si desidera copiare un fotogramma in quelli susseguenti, si può usare la funzione **Copy All**, per evitare di ridisegnare un fotogramma ex novo.

Si può procedere, in seguito, ad un'animazione per modifiche successive di una "traccia" iniziale, nel nostro caso

l'omino fermo del primo fotogramma, che per giunta, nel quinto frame, potrà essere lasciato inalterato.

Tra le funzioni di servizio, notiamo che per cancellare fotogrammi, si può usare **Delete All** (cancella tutti) oppure **Delete Frame**, che si limita ad eliminare solo quello attualmente visualizzato.

Tutto ciò che riguarda l'animazione e lo spostamento nei frames è riportato nel menu **Anim/Control**. Con **Next** e **Previous** ci si può posizionare sul frame successivo e precedente (gli equivalenti sono i tasti **1** e **2**), oppure con **Goto** si può digitare direttamente il numero di frame sul quale posizionarsi per redazioni.

Le due funzioni più importanti per il controllo della riproduzione vera e propria dell'animazione sono comunque **Set Rate** e **Set Frames**. La prima consente di specificare la velocità dell'animazione, espressa in frames al secondo, quindi va regolata in funzione del numero di fotogrammi usati e dell'effetto che si vuole dare.

Con Set Frames si indica se si desidera che l'animazione interessi tutti i fotogrammi (nel nostro caso, da 1 a 7), nel quale caso si clicca su **All Frame**, oppure un certo gruppo **consecutivo** di fotogrammi (ad esempio, **2** e **6**), valori che vanno digitati negli appositi gadget. Cliccando su **Cancel** si fanno ignorare le modifiche, con **OK** le si convalida.

Per eseguire l'animazione vera e propria si può usare **Play**, **Play Once** e **Play Ping Pong**.

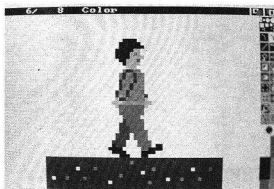
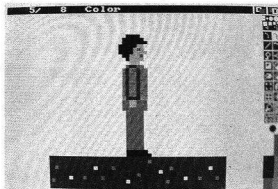
GNel primo caso l'animazione avviene dal primo fotogramma (il primo effettivo o il primo della sequenza parziale specificata con **Range** nel requester **Set Range** dell'opzione di menu **Set Frames**, come già detto) sino all'ultimo fotogramma (assoluto o del Range) ricominciando dal primo. Con **Play Once** l'animazione viene avviene in modo analogo, ma una volta sola; con **Play Ping Pong** viene ripetuta dal primo fotogramma all'ultimo e poi, al rovescio, dall'ultimo al primo.

L'animazione così creata può essere salvata e ricaricata successivamente con le apposite opzioni **Anim/Save** e **Anim/Load**, rispettivamente.

Ultime note

L'animazione può essere salvata su disco per essere ricaricata e riprodotta da DP3 come anche da appositi programmi di pubblico dominio, eseguibili da Shell o da Workbench, come ad esempio **ShowAnim** o **PlayAnim**. Per convenzione, il file di animazione IFF si dovrebbe denominare con il suffisso "**.anim**", in modo da poterlo successivamente riconoscere, come tale, già dal nome.

L'opzione **Anim Brush** consente animazioni in un modo diverso, apparentemente più complicato, basate sulle **brush** e non sulle pagine grafiche. Le sue opzioni, quindi, non sono utili per la tecnica descritta per l'omino che passeggia.

Magari le tratteremo in una futura chiaccherata su CCC, se questa vi è piaciuta e vi è stata utile.

di Tonino Giorgi

CHE CURVE, RAGAZZI!

L'istruzione Circle di AmigaBasic si presta per sviluppare anche (e soprattutto) ellissi; le applicazioni? In Astronomia, ad esempio...

Chi si occupa principalmente di grafica, e si trova spesso a programmare in **AmigaBasic**, avrà certamente dovuto fare i conti, prima o poi, con una dolorosa quanto ingiustificata carenza del suddetto linguaggio.

Ci riferiamo, in particolare, all'istruzione **Circle**, che, come tutti(!) sanno, serve a tracciare **ellissi** (e non solo cerchi, come suggerirebbe il nome) di qualunque forma, dimensione, posizione, colore e altro, ma, ahimè, solo ed invariabilmente con gli assi rigorosamente orizzontali o verticali: una inaccettabile limitazione, non c'è dubbio.

L'insolita lacuna è stranamente diffusa in molti altri interpreti: forse i progettisti dei vari dialetti Basic si dimenticano regolarmente di considerare il fondamentale parametro, o in maniera più superficiale giudicano che sia inconcepibile che l'utente-disegnatore possa mai aver bisogno di disegnare ellissi inclinate; oppure, più brutalmente, se ne fregano.

Un po' di teoria

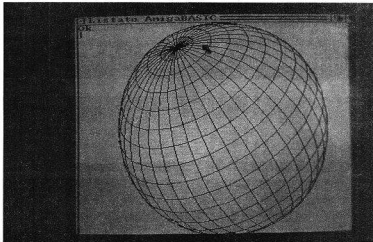
Alla base di una routine che tenga conto dell'inclinazione degli assi c'è, com'è intuibile, della semplice **trigonometria**. La cosa non deve spaventare, si tratta di far lavorare un po' il cervello per accorgersi di quanto siano facili da usare le funzioni **seno**, **coseno**, **tangente** e le loro **inverse**. Fortunatamente non occorre essere ingegneri o architetti per capire le semplici regole di geometria che sono alla base del problema.

La routine

Il sottoprogramma di cui ci occupiamo, denominato **Arc**, serve a tracciare archi d'ellisse; naturalmente può tracciare anche ellissi complete, ma queste, in fondo, sono un caso particolare di arco, in cui l'estremo iniziale e quello finale coincidono.

Dal momento che parliamo di casi particolari, ricordiamo anche, per i più distratti, che la circonferenza è un caso

Sottoprogrammi

La routine è presentata sotto forma di **sottoprogramma**. Per chi non avesse dimestichezza con i sottoprogrammi rimandiamo al capitolo 6 del manuale AmigaBasic ed ai numerosi articoli apparsi su CCC; qui ci limitiamo a ricordare che sono simili alle subroutine, ma al tempo stesso molto più potenti. Possono essere piazzati dovunque e sono veri e propri programmi nel programma, che da questo possono essere "invocati"; di solito utilizzano dei parametri passati al momento della chiamata dal programma stesso; un sottoprogramma inizia con un'istruzione **Sub**, termina con un'istruzione **End Sub**, e viene eseguito con **Call**.

particolare dell'ellisse, nel quale i due assi hanno la stessa lunghezza o, se preferite, in cui l'eccentricità è nulla (qualunque testo scolastico di geometria può comunque rinfrescarvi le idee).

Vediamo ora il significato dei **parametri** che seguono il nome del sottoprogramma. Innanzitutto sarà bene ricordare che le ellissi sono raffigurate avendo come riferimento due assi di simmetria, tra loro ortogonali (uno maggiore e uno minore) nella cui intersezione è posizionato il **centro** dell'ellisse (e non accenniamo ai fuochi perchè la routine non ne fa uso). Per tracciare un'ellisse occorre posizionarla, cioè decidere le coordinate, ad esempio, del centro: a questo servono i primi due parametri, **xcentro** e **ycentro**, che esprimono le coordinate, in pixel, del centro dell'ellisse rispetto ad un punto di riferimento che, per comodità, è bene far coincidere con il **centro dello schermo**. Seguono le misure, sempre in pixel, dei due **semiassi** dell'ellisse, **Semia** (di solito il maggiore) e **Semib** (di solito il minore), che determinano le dimensioni e la "forma" dell'ellisse. Quindi troviamo, come s'intuisce dal nome, il parametro che ha motivato la creazione dell'intera routine: l'**angolo di rotazione** dell'ellisse attorno al suo centro, espresso, come sempre, in radianti.

La rotazione, con angoli positivi, avviene in senso orario, considerando come posizione di partenza, cioè con rotazione nulla, l'asse "**a**" orizzontale (anche se è il minore) e l'asse "**b**" verticale (anche se è il maggiore).

Infine troviamo altri due angoli, **Inizio#** e **Fine#**, che esprimono gli angoli di partenza e d'arrivo dell'arco d'ellisse da tracciare. Vanno misurati in senso orario, in radianti, a partire dall'estremo destro dell'asse "a" (quello orizzontale); naturalmente abbiamo parlato di estremo destro e di asse orizzontale nel caso in cui l'ellisse non sia ruotata. In caso di rotazione l'asse "a" non sarebbe più orizzontale e l'estremo destro potrebbe non essere più tale, ma diventare sinistro (nel caso l'ellisse sia ruotata di un angolo compreso tra 90 e 270 gradi). Si è detto che i due parametri stabiliscono l'inizio e la fine dell'arco; ma se volessimo tracciare un'ellisse intera? Basterebbe porre...

Inizio# = 0

...e...

Fine# = 2 * Pi Greco

O anche: "inizio#" ad un qualunque valore compreso tra 0 e 2 pi greco, e "fine#" al valore di "inizio#" aumentato di 2 pi greco.

Si ricordi inoltre che "fine#" deve essere sempre maggiore di "inizio#".

Dei 7 parametri appena descritti, i primi 5 sono costituiti da numeri in **singola precisione**, mentre gli ultimi due sono espressi da numeri in **doppia precisione**. La distinzione è importante: i 7 valori possono essere, e quasi sempre sono, valori decimali, per cui non si possono usare numeri interi; gli ultimi due richiedono sempre l'impiego di numeri in doppia precisione.

Ciò va tenuto presente nel caso in cui il sottoprogramma venga chiamato avendo, come variabili da passare, numeri di tipo diverso. Per esempio dei numeri interi per un qualunque parametro, o a singola precisione anzichè in doppia per gli ultimi due.

In questo caso dovrete chiamare il sottoprogramma e passare le variabili accompagnandole con un'istruzione che ne cambi il tipo (**CSNG, CDBL**).

Per funzionare, il sottoprogramma Arc richiede altri valori, che, come si vede dalla seconda istruzione (**Shared**), possono essere passati condividendoli con quelli eventualmente già definiti nel programma principale, ma che in alternativa possono essere definiti ex novo.

I valori dei quali stiamo parlando sono: **Pi#**, cioè il valore di pi greco, in doppia precisione; **Aspetto**, per il quale viene usato il valore **1.04**, in funzione del monitor utilizzato.

Infine **Icscentro%** e **Ipsiloncentro%** che esprimono la traslazione orizzontale e verticale dell'origine degli assi cartesiani, e che in pratica, come si può vedere dall'esempio che accompagna il sottoprogramma, la spostano al centro dello schermo.

Routines di Toma, le ricordate?

A proposito del C/64, è proprio da questo computer che nasce il programma pubblicato: oltre ai varii Simons' (e non Simon's come qualcuno si ostina a scrivere) Basic, Ultra Basic e compagnia, chi non ricorda le favolose routine grafiche di Toma? Mentre i primi aggiungevano, allo scarno interprete di serie, una quantità di istruzioni totalmente nuove o modificate, che comunque rendevano molto meno cupa la vita del programmatore, le seconde erano una manciata di istruzioni esclusivamente dedicate alla grafica, che svolgevano a meraviglia il loro compito.

E chi, se non CCC, poteva offrire ai suoi lettori una simile benedizione?

Chi tra i fedelissimi non ricorda il numero 10 e il numero 14 di CCC, e le successive routine aggiuntive? Ah, la gioia che tutti provammo nel poter finalmente disegnare rette e cerchi a volontà sul nostro schermo 320 x 200 senza ricorrere alle famigerate **Poke**! Quante benedizioni e quanta gratitudine al caro **Danilo Toma**, che con la sua notevole opera ha realizzato senza dubbio una pietra miliare nella storia del C/64!

Ma, come succede alle persone pignole, ecco che qualcuno si accorse della mancanza di qualcosa: proprio nelle istruzioni **Circle** e **Arc**, che tracciano ellissi complete o archi di esse. Le suddette coniche risultavano invariabilmente disegnate in posizione verticale ed orizzontale, e non c'era nulla da fare.

Era quindi necessario modificare le pur soddisfacenti routine di Toma individuando la routine l.m. responsabile del calcolo delle coordinate dei punti che costituivano la curva, aggiungendo la lettura del nuovo parametro (l'angolo di rotazione, ovviamente espresso in radianti), e modificando il calcolo dei punti in base all'angolo in questione; il tutto, naturalmente, ancòra in linguaggio macchina.

Viene riproposta, in questa sede, la stessa routine; con la differenza che, invece del complesso linguaggio macchina del C/64, viene utilizzato l'interprete **AmigaBasic**. Con gli stessi risultati e gli stessi vantaggi.

Il pianeta

Un esempio di utilizzo del sottoprogramma Arc è costituito dal listato pubblicato, di nome **Pianeta**.

Si tratta di un ottimo esempio, in quanto fa un uso massiccio di Arc, evidenziandone tutte le possibilità, tracciando sia ellissi complete, sia archi, usando appropriatamente il valore **aspetto** (a patto che lo si sia calcolato in modo preciso) per tracciare sullo schermo cerchi che siano tali e che non appaiano ovali a causa di una (pur minima) differenza di scala sui due assi.

Per calcolare il suddetto valore (metodo un po' rustico, ma efficace), aprite uno schermo 320 x 256 con relativa finestra, come è riportato all'inizio del programma Pianeta, quindi tracciate un cerchio con l'istruzione **Circle (160, 121), 121, , , ,1** dove l'**1** finale rappresenta il valore dell'aspetto.

Quindi prendete un metro, meglio se flessibile, e piazzate il capoccione davanti allo schermo del monitor o TV; chiudendo un occhio cercate di posizionare l'altro esattamente davanti al centro del video.

A questo punto col metro, e senza muovere la testa, misurate il diametro orizzontale del cerchio, e poi quello verticale, al millimetro: se i due valori non sono identici, vuol dire che il valore di "aspetto" non è 1, ma è dato dal diametro orizzontale diviso quello verticale, che fornisce comunque un valore molto vicino all'unità; usate allora il nuovo valore, limitandovi ai primi due decimali, e riprovate: dovreste misurare due diametri identici, ma se così non fosse vorrebbe dire che dovete aggiustare ancora un po' il valore, di qualche centesimo in più o in meno (o andare con urgenza dall'oculista).

Il valore di aspetto così trovato, però, va bene solo usando la bassa risoluzione non interlacciata o l'alta risoluzione interlacciata; negli altri due casi il calcolo è semplice: con l'alta risoluzione non interlacciata il valore da usare sarà la metà di quello calcolato, e viceversa, con la bassa risoluzione interlacciata, se mai la usaste, il valore sarebbe il doppio.

Come funziona

Il programma pubblicato disegna un pianeta rappresentato da un reticolato geografico di meridiani e paralleli; sia gli uni che gli altri vengono disegnati ogni 10 gradi, quindi a partire dall'equatore avremo un parallelo a 10 di latitudine sia nord che sud, uno a 20, nord e sud, uno a 30, e così via fino al Polo; lo stesso per i meridiani: uno ogni 10 di longitudine. Il valore può comunque essere cambiato modificando la variabile **Fascia%** posta all'inizio del programma. Essa è calcolata come 90/9, cioè considerando che in un emisfero, dall'Equatore al Polo, ci sono 90 gradi di latitudine. Dividendo l'emisfero in 9 fasce, avremo un'ampiezza di 10 per ogni fascia; se provate a cambiare il valore 9 in 15, 10, 6 oppure 3 otterrete risultati diversi, ma attenzione: dovrete usare un numero che stia in 90 un numero intero di volte.

Vediamo ora il significato dei due valori da immettere su richiesta del programma: l'**inclinazione** rappresenta l'angolo del quale è ruotato l'asse polare del pianeta rispetto alla posizione verticale: positivo in senso antiorario, e negativo in senso orario. Il valore è espresso in gradi.

La **latitudine** è l'angolo, ancora espresso in gradi, secondo il quale il Polo è rivolto verso di noi: se positivo, vedremo il Polo Nord, se negativo, il Polo Sud. Il motivo per cui abbiamo chiamato latitudine tale angolo è semplice: è come se noi, osservatori, stessimo sospesi sopra la superficie del pianeta, a migliaia di chilometri d'altezza, e guardassimo giù; proprio sotto di noi, in direzione del centro, vedremmo un punto sulla superficie con quella latitudine, punto per il quale noi saremmo allo **Zenit**.

E' un po' come per i satelliti geostazionari: da lassù vedrebbero la Terra con al centro proprio il punto del quale sono allo Zenit (e che ha una certa latitudine).

Potete provare tutte le combinazioni che volete, rispettando i limiti posti dal programma in fase di Input.

I paralleli sono ellissi che hanno la stessa eccentricità, il cui valore è funzione esclusivamente dell'angolo secondo cui il Polo è rivolto verso di noi, che immettiamo in fase di Input e che, come appena spiegato, viene chiamato (forse impropriamente) latitudine.

Per i paralleli cambiano solo le dimensioni che dipendono, stavolta, dalla latitudine, quella vera, cioè la distanza angolare dall'equatore.

Per i meridiani, invece, sono le dimensioni ad essere uguali per tutti, mentre cambiano le eccentricità, che sono funzione sia della longitudine di ogni meridiano, sia del solito angolo chiamato latitudine.

Due parole sulla digitazione dei listati: uno costituisce la routine vera e propria, argomento base di quest'articolo, ed è il sottoprogramma **Arc**; l'altro costituisce solo un esempio d'uso di Arc, ed è il programma **Pianeta**.

Dato che è possibile usare Arc in ogni vostro programma che richieda di disegnare ellissi ruotate, consigliamo la seguente procedura:

digitate soltanto il sottoprogramma Arc e salvatelo digitando, nella finestra di output...

Save "arc", A

...e **non** con **Save** nè con **Save As** del menu!

In seguito, dopo un **New**, digitate il programma Pianeta e, dalla finestra di output, un **Merge "Arc"** riunirà Arc e Pianeta; poi salvate il tutto col nome di Pianeta. In questo modo avrete sempre disponibile la routine **Arc**, da unire in coda ad ogni programma che ne richieda l'uso, ed il programma Pianeta che già la contiene al suo interno.

```
' Esempio d'uso del sottoprogramma ARC: PIANETA
' Linguaggio: AmigaBASIC
' Autore: Tonino Giorgi
' Ascoli Piceno
' Anno di grazia 1990
SCREEN 1, 320, 256, 2, 1: WINDOW 1, , , , 1
pi# = 3.141592653589793#: raggio% = 121: icscentro% = 160: ipsiloncentro% = 121: fascia% = 90 / 9: aspetto = 1.04
' 1.04 e' l'aspetto per lo schermo 3207256 (o 6407512) .
Primo: INPUT"Inclinazione (-180 ... +180) ";incl%: IF ABS (incl%) > 180 THEN Primo
Secondo: INPUT"Latitudine (-90 ... +90) ";lat%: IF ABS (lat%) > 90 THEN Secondo
IF lat% < 0 THEN incl% = incl% - 180: lat% = - lat%
incl = incl% * pi# / 180: lat% = - lat%: lat# = lat% * pi# / 180
coslat = COS (lat#): senlat = SIN (lat#): senlatquad = senlat ^ 2
COLOR 2, 1: CLS: CIRCLE (icscentro%, ipsiloncentro%) , raggio%, , , , aspetto
Paralleli:
FOR fi% = fascia% - 90 TO 90 - fascia% STEP fascia%
fi# = fi% * pi# / 180: discencen = raggio% * SIN (fi#) * coslat
aparall = raggio% * COS (fi#): bparall = aparall * ABS (senlat)
IF lat% = 0 THEN Segmento
IF 90 - ABS (lat%) > fi% AND fi% > ABS (lat%) - 90 THEN Arco
IF lat% > 0 AND 90 > fi% AND fi% > = 90 - ABS (lat%) OR lat% < 0 AND ABS (lat%) - 90 > = fi% AND fi% > - 90 THEN Ellisse
GOTO Prossimo
Segmento: ascissa = raggio% * COS (fi#): ordinata = raggio% * SIN (fi#)
ics = ascissa * COS (incl) / aspetto: ipsilon = - ascissa * SIN (incl)
x1 = icscentro% + discencen * SIN (incl) / aspetto + ics
y1 = ipsiloncentro% + discencen * COS (incl) + ipsilon
x2 = icscentro% + discencen * SIN (incl) / aspetto - ics
y2 = ipsiloncentro% + discencen * COS (incl) - ipsilon
LINE (x1, x2) - (x2, y2): GOTO Prossimo
Arco: abi# = raggio% * SIN (fi#) * senlatquad / coslat
ociquadro# = raggio% ^ 2 * (1 - (SIN (fi#) / coslat) ^ 2)
senteta# = abi# / SQR (abi# ^ 2 + ociquadro#)
teta# = ATN (senteta# / SQR (1 - senteta# ^ 2))
IF teta# < 0 THEN ang1# = teta# + pi# * 2: ang2# = 3 * pi# - teta# ELSE ang1# = teta#: ang2# = pi# - teta#
ARC discencen * SIN (incl) , discencen * COS (incl) , aparall, bparall, incl, ang1#, ang2#: GOTO Prossimo
Ellisse: ARC discencen * SIN (incl) , discencen * COS (incl) , aparall, bparall, incl, 0#, pi# * 2
Prossimo: NEXT
Meridiani:
FOR lambda = fascia% / 2 - 90 TO 90 - fascia% / 2 STEP fascia%
lambda# = lambda * pi# / 180
IF lat% = 0 THEN gamma = 0: bi = raggio% * SIN (lambda#): ang1# = 0#: ang2# = pi#: GOTO Traccia
IF ABS (lat%) = 90 THEN LINE (icscentro% + raggio% * COS (lambda#) / aspetto, ipsiloncentro% + raggio% * SIN (lambda#)) - (icscentro% - raggio% * COS (lambda#) / aspetto, ipsiloncentro% - raggio% * SIN (lambda#)): GOTO Successivo
sengamma = senlat * SIN (lambda#) / SQR (1 - (coslat * SIN (lambda#)) ^ 2)
bi = raggio% * coslat * ABS (sengamma) / SQR (1 - coslat ^ 2 * (1 - sengamma ^ 2))
gamma = ATN (sengamma / SQR (1 - sengamma ^ 2))
ang1# = - pi# * (lambda# > 0): ang2# = pi# + ang1#
Traccia: ARC 0!, 0!, CSNG (raggio%) , bi, incl + gamma + CSNG (pi# / 2) , ang1#, ang2#
Successivo: NEXT

' SOTTOPROGRAMMA ARC

SUB ARC (xcentro, ycentro, semia, semib, angolo, inizio#, fine#) STATIC
SHARED pi#, aspetto, icscentro%, ipsiloncentro%
eccenquadro = 1 - (semib / semia) ^ 2
xcentro = xcentro / aspetto
seno = SIN (angolo): coseno = COS (angolo)
senalfa# = SIN (inizio#) / SQR (1 - eccenquadro * COS (inizio#) ^ 2)
senbeta# = SIN (fine#) / SQR (1 - eccenquadro * COS (fine#) ^ 2)
alfa# = ATN (senalfa# / SQR (1 - senalfa# ^ 2))
IF COS (inizio#) < 0 THEN alfa# = pi# - alfa# ELSE IF alfa# < 0 THEN alfa# = 2 * pi# + alfa#
beta# = ATN (senbeta# / SQR (1 - senbeta# ^ 2))
IF COS (fine#) < 0 THEN beta# = pi# - beta# ELSE IF beta# < 0 THEN beta# = 2 * pi# + beta#
IF fine# > 2 * pi# THEN beta# = beta# + 2 * pi#
ampiezza# = beta# - alfa#
lati% = INT (ampiezza# / pi# * 20): passo# = ampiezza# / lati%
ro = ABS (semib) / SQR (1 - eccenquadro * COS (inizio#) ^ 2)
ics = ro * COS (inizio#): ipsilon = ro * SIN (inizio#)
x1 = (ics * coseno + ipsilon * seno) / aspetto
y1 = ipsilon * coseno - ics * seno
FOR gamma% = 1 TO lati%: gamma# = alfa# + gamma% * passo#
ascissa = semia * COS (gamma#): ordinata = semib * SIN (gamma#)
x2 = (ascissa * coseno + ordinata * seno) / aspetto
y2 = ordinata * coseno - ascissa * seno
LINE (icscentro% + xcentro + x1, ipsiloncentro% + ycentro + y1) - (icscentro% + xcentro + x2, ipsiloncentro% + ycentro + y2)
x1 = x2: y1 = y2
NEXT
END SUB
```

GUIDA ALL'ACQUISTO

QUANTO COSTA IL TUO COMMODORE

Amiga 2000 - L. 2.715.000

Microprocessore Motorola MC68000 - Clock 7.16MHz - Kickstart ROM - Memoria RAM: 1 MByte - 3 chip custom per DMA, Video, Audio, I/O - 5 Slot di Espansione Amiga Bus 100 pin AutoconfigTM - 1 Slot di Espansione 86 pin per Schede Coprocessore - 2 Slot di Espansione compatibili AT/XT - 2 Slot di Espansione compatibili XT - 2 Slot di Espansione Video - 1 Floppy Disk Drive da 3 1/2", 880 KBytes - Porta seriale RS232C - Sistema Operativo single-user, multitasking AmigaDOS - Compatibilità MS-DOS XT/AT disponibile con schede interne Janus (A2088 - A2286) - Monitor escluso

Amiga 500 - L. 995.000

Microprocessore Motorola MC68000 - Clock 7.16 MHz - Kickstart ROM - Memoria RAM: 512 KBytes - 3 Chip custom per DMA, Video, Audio, I/O - 1 Floppy Disk Driver da 3 1/2", 880 KBytes - Porta seriale RS232C - Porta parallela Centronics

Videomaster 2995 - L. 1.200.000

Desk Top Video - Sistema per elaborazini video semiprofessionale composto da genlock, digitalizzatore e alloggiamento per 3 drive A2010 - Ingressi videocomposito (2), RGB - Uscite Videocomposito, RF, RGB + sync -

Floppy Disk Driver A 1010 - L. 335.000

Floppy Disk Driver - Drive esterno da 3 1/2" - Capacità 880 KBytes - Collegabile a tutti i modelli della linea Amiga, alla scheda A2088 e al PC1

Floppy Disk Drive A 2010 - L. 280.000

Floppy Disk Drive - Drive interno aggiuntivo da 3 1/2" - Capacità 880 KBytes - Collegabile ad Amiga 2000

Hard Disk A 590 - L. 1.750.000

Hard Disk+Controller+RAM - Scheda Controller - Hard Disk da 3 1/2" 20 MBytes - 2 MBytes "fast" RAM - Collegabile all'Amiga 500

Scheda Janus A 2088 + A 2020 - L. 1.050.000

Scheda Janus XT+Floppy Disk Drive da 5 1/4", 360 KBytes - Scheda Bridgeboard per compatibilità MS-DOS (XT) in Amiga 2000 - Microprocessore Intel 8088 - Coprocessore matematico opzionale Intel 8087

A2286+A2020 - L. 1.985.000

Scheda Janus AT+Floppy Disk Drive da 5 1/4", 1.2 MBytes - Scheda Bridgeboard per compatibilità MS-DOS (AT) in Amiga 2000 - Microprocessore Intel 80287 - Clock 8 MHz - RAM: 1 MBytes on-board - Floppy Disk Controller on-board - Floppy Disk Driver disegnato per l'installazione all'interno dell'Amiga 2000 -

Scheda A2620 - L. 2.700.000

Scheda Processore Alternativo 32 bit - Scheda per 68020 e Unix - Microprocessore Motorola MC68020 - Coprocessore matematico Motorola MC68881 (opzionale MC68882)

Scheda A Unix - L. 3.250.000

Sistema Operativo AT&T Unix System V Release 3 - Per Amiga 2000 con scheda A2620 e Hard Disk 100 MBytes

Hard Disk A2092+PC5060 - L. 1.020.000

Hard Disk e controller - Hard Disk 3 1/2" ST506 - Capacità formattata 20 MBytes

Hard Disk A2090+2092 - L. 1.240.000

Hard Disk e controller - Hard Disk 3 1/2" ST506 - Capacità formattata 20 MBytes

Hard Disk A2090+A2094 - L. 1.900.000

Stesse caratteristiche del kit A2092 ma con disco da 40 MBytes

Espansione di memoria A2058 - L. 1.149.000

Espansione di memoria - Scheda di espansione per Amiga 2000 - Fornita con 2 MBytes "fast" RAM, espandibile a 4 o 8 MBytes

Scheda Video A2060 - L. 165.000

Modulatore video - Scheda modulatore video interna per Amiga 2000 - Uscite colore e monocromatica - Si inserisce nello slot video dell'Amiga 2000

Genlock Card A2301 - L. 420.000

Genlock - Scheda Genlock semiprofessionale per Amiga 2000 - Permette di miscelare immagini provenienti da una sorgente esterna con immagini provenienti dal computer

Professional Video Adapter Card A2351 - L. 1.500.000

Professional Video Adapter - Scheda Video Professionale per Amiga 2000 (B) - Genlock qualità Broadcast - Frame Grabber - Digitalizzatore - Include software di controllo per la gestione interattiva
(Disponibile da maggio '89)

A501 - L. 300.000

Espansione di memoria - Cartuccia di espansione di memoria da 512 KBytes per A500

A520 - L. 45.000

Modulatore RF - Modulatore esterno A500 - Permette di connettere qualsiasi televisore B/N o colori ad Amiga 500

A Scart - L. 27.000

Cavo di collegamento A500/A2000 con connettore per televisione SCART

Monitor a colori 1084 - L. 595.000

Monitor a colori ad alta risoluzione - Tubo 14" Black Matrix antiriflesso - Pitch 0.39 mm - Compatibile con Amiga 500/2000, PC (tutta la gamma), C64 e C128

Monitor a colori 2080 - L. 770.000

Monitor a colori ad alta risoluzione e lunga persistenza - Tubo 14" Black Matrix antiriflesso - Pitch 0.39 mm - Frequenza di raster 50 Hz - Compatibile con Amiga 500/2000, PC (tutta la gamma), C64 e C128

Monitor Monocromatico A2024 - L. 1.235.000

Monitor monocromatico a fosfori "bianco-carta" - Turbo 14" antiriflesso - (Disponibile da marzo '89)

PC60/40 - L. 7.812.000

Microprocessore Intel 80386 - Coprocessore matematico opzionale Intel 80387 - Clock 8 o 16 MHz selezionabile via software e da tastiera - Monitor monocromatico 14" - Tastiera avanzata 102 tasti con 12 funzioni - Sistema Operativo MS-DOS 3 2.1 - Interprete GW-Basic

PC60/40C - L. 8.127.000

Stessa configurazione ma con monitor 14" a colori mod. 1084

PC 60/80 - L. 10.450.000

Microprocessore Intel 80386 - Coprocessore opzionale Intel 80387 - Clock 8 o 16 MHz selezionabile via software e da tastiera - Memoria RAM: 2.5 MBytes - 1 Floppy Disk Drive da 5 1/4", 1.2 MBytes - 1 Floppy Disk Drive opzionale da 3 1/2", 1.44 MBytes - 1 Hard Disk da 80 MBytes - 2 Porte parallele Centronics - Mouse video EGA (compatibile MDA - Hercules - CGA). Emulazioni disponibili via hardware e software - Monitor monocromatico 14" - Tastiera avanzata 102 tasti con 12 tasti funzione - Sistema Operativo MS-DOS 3.21 - Ambiente Operativo Microsoft Windows/386 - Interprete GW-Basic

PC60/80C - L. 10.700.000

Stessa configurazione ma con monitor 14" a colori mod. 1084

PC40/20 - L. 4.100.000

Microprocessore Intel 80286 - Coprocessore matematico opzionale Intel 80287 - Clock 6 o 10 MHz selezionabile via software, hardware o da tastiera - Memoria RAM: 1 MByte - 1 Floppy Disk Drive da 5 1/4", 1.2 MBytes - 1 Hard Disk da 20 MBytes - Porta seriale RS232 - Porta parallela Centronics - Scheda video AGA multistandard (MDA - Hercules - CGA) Emulazioni disponibili via hardware e software - Monitor monocromatico 14" - Tastiera avanzata 102 tasti con 12 tasti funzione - Sistema Operativo MS-DOS 3.21 - Interprete GW-Basic

PC40/20C - L. 4.350.000

Stessa configurazione ma con monitor 14" a colori mod. 1084

PC 40/40 - L. 5.285.000

Microprocessore Intel 80286 - Coprocessore matematico opzionale Intel 80287 - Clock 6 o 10 MHz selezionabile via software, hardware o da tastiera - Memoria RAM: 1 MByte - 1 Floppy Disk Drive da 5 1/4", 1.2 MBytes - 1 Hard Disk da 20 MBytes - Porta seriale RS232 - Porta parallela Centronics - Scheda video AGA multistandard (MDA - Hercules - CGA) Emulazioni disponibili via hardware e software - Monitor monocromatico 14" - Tastiera avanzata 102 tasti con 12 tasti funzione - Sistema Operativo MS-DOS 3.21 - Interprete GW-Basic

PC40/40C - L. 5.535.000

Stessa configurazione ma con monitor 14" a colori mod. 1084

1352 - L. 78.000

Mouse - Collegabile con Microsoft Bus Mouse - Collegabile direttamente a PC1, PC10/20 - III, PC40 - III

PC910 - L. 355.000

Floppy Disk Drive - Drive interno aggiuntivo da 3 1/2" per PC10/20-I-II-III - Capacità 360 o 720 KBytes selezionabile tramite "config. sys" - Corredo di telaio di supporto per l'installazione in un alloggiamento per un drive da 5 1/4" - Interfaccia identica ai modelli da 5 1/4"

PC1 - L. 995.000

Microprocessore Intel 8088 - 1 Floppy Disk Drive da 5 1/4" - Porta seriale RS232C - Porta parallela Centronics - - Monitor monocromatico 12" - Tastiera 84 tasti - Sistema Operativo MS-DOS 3.2 - Interprete GW-Basic

PCEXP1 - L. 640.000

PC Expansion Box - Box esterno di espansione per PC 1 - Alimentatore aggiuntivo incluso - Contiene 3 Slot di Espansione compatibili Ibm XT - Alloggiamento per Hard Disk da 5 1/4" - Si posiziona sotto il corpo del PC1 e viene collegato tramite degli appositi connettori

PC10-III - L. 1.360.000

Microprocessore Intel 8088 Clock 4.77 MHz 9.54 MHz (double) selezionabile via software e da tastiera - Memoria RAM: 640 KBytes - 2 Floppy Disk Drive da 5 1/4", 360 KBytes - Porta seriale RS232C - Porta parallela Centronics - Porta Mouse per Mouse Commodore 1352 (compatibile Microsoft Bus Mouse - Tastiera avanzata 102 con 12 tasti funzione Sistema Operativo MS-DOS 3.21 - Interprete GW-Basic

PC10-IIIC - L. 1.675.000

Stessa configurazione ma con monitor 14" a colori mod. 1084

PC20-III - L. 2.095.000

Microprocessore Intel 8088 - Clock 4.77 MHz 9.54 MHz (double) selezionabile via software e da tastiera - 1/4", 360 KBytes - 1 Hard Disk da 20 MBytes - Porta seriale RS232C - Porta parallela Centronics - Porta Mouse per Mouse Commodore 1352 (compatibile Microsoft Bus Mouse) - Tastiera avanzata 102 con 12 tasti funzione Sistema Operativo MS-DOS 3.21 - Interprete GW-Basic

PC20-IIIC - L. 2.410.000
Stessa configurazione ma con monitor 14" a colori mod. 1084

Nuovo C64 - L. 325.000
Nuovo Personal Computer CPU 64 KBytes RAM - Vastissima biblioteca software disponibile - Porta seriale Commodore - Porta registratore per cassette - Porta parallela programmabile -

C128D - L. 895.000
Personal Computer CPU 128 KBytes RAM espandibile a 512 KBytes - ROM 48 KBytes - Basic 7.0 - Tastiera separata - Funzionante in modo 128,64 o CP/M 3.0 - Include floppy disk drive da 340 KBytes

Floppy Disk Drive 1541 II - L. 365.000
Floppy Disk Drive - Floppy Disk Drive da 5 1/4" singola faccia - Capacità 170 KBytes - Alimentazione separata - Compatibile con C64, C128, C128D

Floppy Disk Dirve 1581 - L. 420.000
Floppy Disk Drive da 3 1/2" doppia faccia - Capacità 800 KBytes - Alimentazione separata - Compatibile con C64, C128, C128D

1530 - L. 55.000
Registratore a cassette per C64, C128, C128D

Accessori per C64 - 128D
1700 - Espansione di memoria - Cartuccia di espansione di memoria a 128 KBytes per C128 **- L. 170.000**

1750 - Espansione di memoria - Cartuccia di espansione di memoria 512 KBytes per C128 - **L. 245.000**

1764 - Espansione di memoria - Cartuccia di espansione di memoria a 256 KBytes per C64 - Fornita di alimentatore surdimensionato - **L. 198.000**

16499 - Adattatore Telematico Omologato - Collegabile al C64 - Permette il collegamento a Videotel, P.G.E. e banche dati **L. 149.000**

1399 - Joystick - Joystick a microswitch con autofire - **L. 29.000**

1351 - Mouse - Mouse per C64, C128, C128D - **L. 72.000**

Monitor Monocromatico 1402 - L. 280.000
Monitor monocromatico a fosfori "bianco-carta" - Turbo 12" antiriflesso - Ingresso TTL - Compatibile con tutta la gamma PC

Monitor Monocromatico 1404 - L. 365.000
Monitor monocromatico a fosfori ambra - Turbo 14" antiriflesso a schermo piatto - Ingresso TTL - Compatibile con tutta la gamma PC - Base orientabile

Monitor Monocromatico 1450 - L. 470.000
Monitor monocromatico BI-SYNC a fosfori "bianco-carta" - Turbo 14" antiriflesso - Ingresso analogico e digitale - Doppia frequenza di sincronismo orizzontale per compatibilità con adattatori video MDA, Hercules, CGA, EGA e VGA

Monitor a colori 1802 - L. 445.000
Monitor a colori - Turbo 14" - Collegabile a C64, C128, C128D

Monitor monocromatico 1900 - L. 199.000
Monitor monocromatico a fosfori verdi - Turbo 12" antiriflesso - Ingresso videocomposito - Compatibile con tutta la gamma Commodore

Monitor a colori 1950 - L. 1.280.000
Monitor a colori BI-SYNC alta risoluzione - Turbo 14" antiriflesso - Ingresso analogico e digitale - Doppia frequenza di sincronismo orizzontale per compatibilità con adattatori video MDA, Hercules, CGA, EGA e VGA

Stampante MPS 1230 - L. 465.000
Stampante a matrice di punti - Testina a 9 aghi - 120 cps - Bidirezionale - 80 colonne - Near Letter Quality - Stampa grafica - Fogli singoli e modulo continuo - Trascinamento a trattore e/o frizione - Interfaccia seriale Commodore e parallela Centronics - Compatibile con tutti i prodotti Commodore

MPS 1230R - L. 19.000
Nastro per stampante

Stampante MPS 1500C - L. 495.000
Stampante a colori a matrice di punti - Testina a 9 aghi - 130 cps - Bidirezionale - 80 colonne - Supporta nastro a colori o nero - Near Letter Quality - Stampa grafica - Fogli singoli e modulo continuo - Trascinamento a trattore e/o frizione - Interfaccia parallela Centronics - Compatibile con la gamma Amiga e PC

MPS1500R - L. 37.000
Nastro a colori per stampante

MPS1500R - L.37.000
Nastro a colori per stampante

Stampante MPS 1550C - L. 575.000
Stampante a colori a matrice di punti - Testina a 9 aghi - 130 cps - Bidirezionale - 80 colonne - Supporta nastro a colori o nero - Near Letter Quality - Stampa grafica - Fogli singoli e modulo continuo - Trascinamento a trattore e/o frizione - Interfaccia seriale Commodore e parallela Centronics - Compatibile con tutti i prodotti Commodore

I COMMODORE POINT

LOMBARDIA

Milano
- AL RISPARMIO - V.LE MONZA 204
- BCS - VIA MONTEGANI 11
- BRAHA A. - VIA PIER CAPPONI 5
- E.D.S. - C.SO PORTA TICINESE 4
- FAREF - VIA A. VOLTA 21
- FLOPPERIA - V.LE MONTENERO 31
- GBC - VIA CANTONI 7 - VIA PETRELLA 6
- GIGLIONI - V.LE LUIGI STURZO 45
- L'UFFICIO 2000 - VIA RIPAMONTI 213
- LOGITEK - VIA GOLGI 60
- LU - MEN - VIA SANTA MONICA 3
- MARCUCCI - VIA F.LLI BRONZETTI 37
- MELCHIONI - VIA P. COLLETTA 37
- MESSAGGERIE MUSICALI - GALLERIA DEL CORSO 2
- NEWEL - VIA MAC MAHON 75
- PANCOMMERZ ITALIA - VIA PADOVA 1
- SUPERGAMES - VIA VITRUVIO 38
- 68000 E DINTORNI - VIA WASHINGTON 91

Provincia di Milano
- GINO FERRARI CENTRO HI-FI - VIA MADRE CABRINI 44 - S. ANG. LODIGIANO
- F.LLI GALIMBERTI - VIA NAZIONALE DEI GIOVI 28/36 - BARLASSINA
- TECNOLUX - VIA PIETRO NENNI 5 - BERNATE TICINO
- OGGIONI & C. - VIA DANTE CESANA 27 - CARATE BRIANZA
- AL RISPARMIO - VIA U. GIORDANO 57 - CINISELLO BALSAMO
- GBC - V.LE MATTEOTTI 66 - CINISELLO BALSAMO
- CASA DELLA MUSICA - VIA INDIPENDENZA 21 - COLOGNO MONZESE
- PENATI - VIA VERDI 28/30 - CORBETTA
- EPM SYSTEM - V.LE ITALIA 12 - CORSICO
- P.G. OSTELLARI - VIA MILANO 300 - DESIO
- CENTRO COMPUTER PANDOLFI - VIA CORRIDONI 18 - LEGNANO
- COMPUTEAM - VIA VECELLIO 41 - LISSONE
- M.B.M. - C.SO ROMA 112 - LODI
- L'AMICO DEL COMPUTER - VIA CASTELLINI 27 - MELEGNANO
- BIT 84 - VIA ITALIA 4 - MONZA
- IL CURSORE - VIA CAMPO DEI FIORI 35 - NOVATE MIL.
- I.C.O. - VIA DEI TIGLI 14 - OPERA
- R & C ELGRA - VIA SAN MARTINO 13 - PALAZZOLO MIL.
- ESSEGIEMME SISTEMI SAS - VIA DE AMICIS 24 - RHO
- TECNO - CENTRO - VIA BARACCA 2 - SEREGNO
- NIWA HARD&SOFT - VIA B. BUOZZI 94 - SESTO SAN GIOV.
- COMPUTER SHOP - VIA CONFALONIERI 35 - VILLASANTA
- ACTE - VIA B. CREMIGNANI 13 - VIMERCATE
- IL COMPUTER SERVICE SHOP - VIA PADANA SUPERIORE 197 - VIMODRONE

Bergamo
- D.R.B. - VIA BORGO PALAZZO 65
- TINTORI ENRICO &C. - VIA BROSETA 1
- VIDEO IMMAGINE - VIA CARDUCCI c/o CITTA' DI MERCATO

Provincia di Bergamo
- BERTULEZZI GIOVANNI - VIA FANTONI 48 - ALZANO LOMBARDO
- COMPUTER SHOP - VIA VITTORIO VENETO 9 - CAPRIATE SAN GERVASIO
- B M R - VIA BUTTARO 4/T - DALMINE
- MEGABYTE 2 - VIA ROMA 61/A - GRUMELLO
- OTTICO OPTOMETRISTA ROVETTA - P.ZZA GARIBALDI 6 - LOVERE
- COMPUTER POINT - VIA LANTIERI 52 - SARNICO
- A.B. INFORMATICA - STRADA STATALE CREMASCA 66 - URGNANO

Brescia
- MASTER INFORMATICA - VIA F.LLI UGONI 10/B

PROVINCIA DI BRESCIA
- MISTER BIT - VIA MAZZINI 70 - BRENO
- CAVALLI PIETRO - VIA 10 GIORNATE 14 BIS - CASTREZZATO
- VIETTI GIUSEPPE - VIA MILANO 1/B - CHIARI
- MEGABYTE - P.ZZA MALUEZZI 14 - DESENZANO DEL GARDA
- BARESI RINO &C. - VIA XX SETTEMBRE 7 - GHEDI
- INFO CAM - VIA PROVINCIALE 3 - GRATACASOLO
- "PAC-LAND" di GARDONI - CENTRO COM.le - LA CASA DI MARGHERITA D'ESTE - VIA GIORGIONI 21

Como
- IL COMPUTER - VIA INDIPENDENZA 90
- 2M ELETTRONICA - VIA SACCO 3

Provincia di Como
- ELTRON - VIA IV NOVEMBRE 1 - BARZANO'
- DATA FOUND - VIA A. VOLTA 4 - ERBA
- CIMA ELETTRONICA - VIA L. DA VINCI 7 - LECCO
- FUMAGALLI - VIA CAIROLI 48 - LECCO
- RIGHI ELETTRONICA - VIA G. LEOPARDI 26 - OLGIATE COMASCO

Cremona
- MONDO COMPUTER - VIA GIUSEPPINA 11/B
- PRISMA - VIA BUOSO DA DOVARA 8
- TELCO - P.ZZA MARCONI 2/A

Provincia di Cremona
- ELCOM - VIA IV NOVEMBRE 56/58 - CREMA
- EUROELETTRONICA - VIA XX SETTEMBRE 92/A - CREMA

Mantova
- COMPUTER CANOSSA - GAL. FERRI 7
- 32 BIT - VIA C. BATTISTI 14
- ELET. di BASSO - V.LE RISORGIMENTO 69

Provincia di Mantova
- CLICK - ON COMPUTER - S.S. GOITESE 168 - GOITO

Pavia
- POLIWARE - C.SO C. ALBERTO 76
- SENNA GIANFRANCO - VIA CALCHI 5

Provincia di Pavia
- A. FERRARI - C.SO CAVOUR 57 - MORTARA
- LOGICA MAINT - V.LE M.TE GRAPPA 32 - VIGEVANO
- M. VISENTIN - C.SO V. EMANUELE 76 - VIGEVANO

Sondrio
- CIPOLLA MAURO - VIA TREMOGGE 25

Provincia di Sondrio
- FOTONOVA - VIA VALERIANA 1 - S.PIETRO DI BERBENNO

Varese
- ELLE - EFFE - VIA GOLDONI 35
- IL C.TRO ELET. - VIA MORAZZONE 2
- SUPERGAMES - VIA CARROBBIO 13

Provincia di Varese
- BUSTO BIT - VIA GAVINANA 17-BUSTO A.
- MASTER PIX - VIA S.MICHELE 3 - BUSTO A.
- PUNTO UFFICIO - VIA R.SANZIO 8 - GALLARATE
- GRANDI MAGAZZINI BOSSI - VIA CLERICI 196 - GERENZANO
- J.A.C. - C.so MATTEOTTI 38 - SESTO C.

PIEMONTE

Alessandria
- BIT MICRO - VIA MAZZINI 102
- SERV. INFOR. - VIA ALESSANDRO III 47

Provincia di Alessandria
- SONY ITALIANA - VIA G. MANARA 7 - CASALE MONFERRATO
- SGE ELETTRONICA - VIA BANDELLO 19 - TORTONA
- COMPUTER TEMPLE - VIA F. CAVALLOTTI 13 - VALENZA

Asti
- ASTI GAMES - C.SO ALFIERI 26
- RECORD - C.SO ALFIERI 166/3 (Galleria Argenta)

Cuneo
- ROSSI COMPUTERS - C.SO NIZZA 42

Provincia di Cuneo
- PUNTO BIT - C.SO LANGHE 26/C - ALBA
- BOSETTI - VIA ROMA 149 - FOSSANO
- COMPUTERLAND - VIA MAZZINI 30/32 - SALUZZO

Novara
- PROGRAMMA 3 - V.LE BUONARROTI 8
- PUNTO VIDEO - C.so RISORGIMENTO 39/B

Provincia di Novara
- COMPUTER - VIA MONTE ZEDA 4 - ARONA
- ALL COMPUTER - C.SO GARIBALDI 106 - BORGOMANERO
- S.P.A. - C.SO DISSEGNA 21/BIS - DOMODOSSOLA
- ELLIOTT COMPUTER SHOP - VIA DON MINZONI 32 - INTRA
- TRISCONI VALERIA - VIA MAZZINI 90 - OMEGNA

Torino
- ABA ELETTRONICA - VIA C. FOSSATI 5/P
- ALEX COMPUTER E GIOCHI - C.SO FRANCIA 333/4
- COMPUTER HOME - VIA SAN DONATO 46/D
- COMPUTING NEW - VIA M. POLO 40/E
- C.D.M. ELETTR. - VIA MAROCHETTI 17
- DE BUG - C.SO V. EMANUELE II 22
- DESME UNIVERSAL - VIA S.SECONDO 95
- FDS ALTERIO - VIA BORGARO 86/D
- IL COMPUTER - VIA N. FABRIZI 126
- MICRONTEL - C.SO D. degli ABRUZZI 28
- PLAY GAMES SHOP - VIA C. ALBERTO 39/E
- RADIO TV MIRAFIORI - C.SO UNIONE SOVIETICA 381
- SMT ELETTRONICA - VIA BIBIANA 83/bis

Provincia di Torino
- PAUL E CHICO VIDEOSOUND - VIA V.EMANUELE 52 - CHIERI
- BIT INFORMATICA - VIA V. EMANUELE 154 - CIRIE'
- HI - FI CLUB - C.SO FRANCIA 92C - COLLEGNO
- MISTER PERSONAL - VIA CATTANEO 52 - FAVRIA
- I.C.S. - VIA TORINO 73 - IVREA
- DAG - VIA I MAGGIO 40 - LUSERNA S. GIOVANNI
- EUREX - C.SO INDIPENDENZA 5 - RIVAROLO CANAVESE
- DIAM INFORMATICA - C.SO FRANCIA 146 bis - RIVOLI
- FULLINFORMATICA - VIA V.VENETO 25 - RIVOLI
- GAMMA COMPUTER - VIA CAVOUR 3A-3B - SET.TORINESE

Vercelli
- ELETTROGAMMA - C.SO BORMIDA 27 ang. V.Montanara
- ELETTRONICA - STRADA TORINO 15

Provincia di Vercelli
- C.S.I. TEOREMA - VIA LOSANA 9 - BIELLA
- SIGEST - VIA BERTODANO 8 - BIELLA
- REMONDINO FRANCO - VIA ROMA 5 - BORGOSESIA
- FOTOSTUDIO TREVISAN - VIA XXV APRILE 24/B - COSSATO
- STUDIO FOTOGRAFICO IMARISIO - P.ZZA M. LIBERTA' 7 - TRINO

VENETO

Belluno
- UP TO DATE - VIA V. VENETO 43

Provincia di Belluno
- GUERRA COMPUTERS - V.LE MAZZINI 10/A - FELTRE

Padova
- BIT SHOP - VIA CAIROLI 11
- COMPUMANIA - VIA T. CAMPOSANPIERO 37
- D.P.R. DE PRATO R. - V.LO LOMBARDO 4
- G.F. MARCATO - VIA MADONNA DELLA SALUTE 51/53
- SARTO COMPUTER - VIA ARMISTIZIO 79

Provincia di Padova
- COMPUTER SERVICE - BORGO TREVISO 150 - CITTADELLA

Treviso
- BIT 2000 - VIA BRANDOLINI D'ADDA 14
- GUERRA EGIDIO & C. - V.LE CAIROLI 95

Provincia di Treviso
- DE MARIN COMPUTERS - VIA MATTEOTTI 142 - CONEGLIANO
- SIDESTREET - VIA SALVO D'ACQUISTO 8 - MONTEBELLUNA
- FALCON ELETTROAUDIOVIDEO - VIA TERRAGGIO 116 - PREGANZIOL

Venezia
- GUERRA EGIDIO & C. - VIA BISSUOLA 20/A - MESTRE
- TELERADIO FUGA - SAN MARCO 3457

Provincia di Venezia
- GUERRA EGIDIO & C. - VIA VIZZOTTO 29 - SAN DONA' DI PIAVE
- REBEL - VIA F. CRISPI 10 - SAN DONA' DI PIAVE

Verona
- CASA DELLA RADIO - VIA CAIROLI 10
- TELESAT - VIA VASCO DE GAMA 8

Provincia di Verona
- UBER - CP 0363(RAG.SOC. DERTA) - VIA MASCAGNI 31 - CASTEL D'AZZANO
- FERRARIN - VIA DEI MASSARI 10 - LEGNAGO
- COMPUTERS CENTER - VIA CANTORE 26 - VILLAFRANCA

Vicenza
- ELET. BISELLO - V.LE TRIESTE 427/429
- SCALCHI MARKET - VIA C.A' BALBI 139

Provincia di Vicenza
- SCHIAVOTTO - VIA ZANELLA 21 - CAVAZZALE
- GUERRA E. & C. - V.LE DELLE INDUSTRIE - MONTECCHIO MAGGIORE

FRIULI VENEZIA GIULIA

Gorizia
- E.CO. ELETTRONICA - VIA F.LLI COSSAR 23

Trieste
- AVANZO GIACOMO - P.ZZA CAVANA 7
- COMPUTER SHOP - VIA P. RETI 6
- COMPUTIGI - VIA XX SETTEMBRE 51
- CTI - VIA PASCOLI 4

Udine
- MOFERT 2 - VIA LEOPARDI 21
- R.T. SISTEM UDINE - VIA L. DA VINCI 99

Provincia di Udine
- IL PUNTO ELETTRONICO - VIA VENDRAMIN 184 - LATISANA
- IDRENO MATTIUSSI &C. - VIA LICINIANA 58 - MARTIGNACCO

TRENTINO ALTO ADIGE

Bolzano
- COMPUTER POINT - VIA ROMA 82/A
- MATTEUCCI PRESTIGE - VIA MUSEO 54

Provincia di Bolzano
- RADIO MAIR-ELECTRO - VIA CENTRALE 70 - BRUNICO
- ELECTRO RADIO HENDRICH - VIA DELLE CORSE 106 - MERANO
- ERICH KONTSCHIEDER - PORTICI 313 - MERANO
- ELECTRO TAPPEINER - P.ZZA PRINCIPALE 90 - SILANDRO

Trento
- CRONST - VIA G. GALILEI 25

Provincia di Trento

• AL RISPARMIO - C.SO VERONA 138 - ROVERETO

LIGURIA
Genova
• ABM COMPUTER - P.ZZA DE FERRARI 24 rosso
• CAPRIOTTI G. - IA MAMIANI 4r - SAMPIERDARENA
• C.tro ELET. - VIA CHIARAVAGNA 10 R - VIA SESTRI 69R
• COM.le SOTTORIPA - VIA SOTTORIPA 115/117
• FOTOMONDIAL - VIA DEL CAMPO 3-5-9-11-13 r
• LA NASCENTE - VIA SAN LUCA 4/1
• PLAY TIME - VIA GRAMSCI 3/5/7 rosso
• RAPPR-EL - VIA BORGORATTI 23 R
Imperia
• CASTELLINO - VIA BELGRANO 44
Provincia di Imperia
• CENTRO HI-FI VIDEO - VIA DELLA REPUBBLICA 38 -SANREMO
• CASTELLINO - VIA GENOVA 48 - VENTIMIGLIA
La Spezia
• I.L. ELETTRONICA - VIA V. VENETO 123
Provincia di La Spezia
• I.L. ELETTRONICA - VIA AURELIA 299 - FORNOLA DI VEZZANO
Savona
• CASTELLINO - C.SO TARDY E BENECH 101
Provincia di Savona
• CELESIA ENZA - VIA GARIBALDI 144 - LOANO

EMILIA
Bologna
• EUROELETTRICA - VIA RANZANI 13/2
• MINNELLA ALTA FEDELTA' - VIA MAZZINI 146/2
• MORINI & FEDERICI - VIA MARCONI 28/C
• STERLINO - VIA MURRI 73/75
Provincia di Bologna
• S.C. COMPUTERS - VIA E. FERMI 4 - CASTEL SAN PIETRO
• S.P.E. INFORMATICA - VIA DI MEZZO PONENTE 385 - CREVALCORE
• ARCHIMEDE SISTEMI - VIA EMILIA 124 - S. LAZZARO DI SAVENA
Modena
• CO - EL - VIA CESARI 7
• ORSA MAGGIORE - P.ZZA MATTEOTTI 20
• VIDEO VAL WILLY COMPUTERS - VIA CANALETTO 223
Provincia di Modena
• NEW MEDIA SYSTEM - VIA ROMA 281 - SOLIERA
Parma
• BABARELLI G. - VIA B. PARENTE 14/A/B
Provincia di Parma
• PONGOLINI - VIA CAVOUR 32 - FIDENZA
Piacenza
• COMPUTER LINE - VIA G. CARDUCCI 4
• DELTA COMPUTER - VIA M. DELLA RESISTENZA 15/G

TEGGIO EMILIA
• COMPUTERLINE - VIA SAN ROCCO 10/C
• POOL SHOP - VIA EMILIA S. STEFANO 9/C
Provincia di Reggio Emilia
• MACCHIONI - VIA STATALE 467 - CASALGRANDE
ROMAGNA
Ferrara
• BUSINESS POINT - VIA CARLO MAYER 85
Forlì
• COMPUTER VIDEO CENTER - VIA CAMPO DI MARTE 122
Provincia di Forlì
• TOP BIT - VIA VENETO 12 - FORLIMPOPOLI
• COMPUTER HOUSE - V.LE TRIPOLI 193/D - RIMINI
• EASY COMPUTER - VIA LAGOMAGGIO 50 - RIMINI

REPUBBLICA S. MARINO
Ravenna
• COMPUTER HOUSE - VIA TRIESTE 134
Provincia di Ravenna
• ARGNANI - P.ZZA DELLA LIBERTA' 5/A - FAENZA
• ELECTRON INFORMATICA - VIA F.LLI CORTESI 17 - LUGO
• P.L.Z. INFORMATICA - P.ZZA SERCOGNANI 6 - FAENZA

TOSCANA
Arezzo
• DELTA SYSTEM - VIA PIAVE 13
Firenze
• ATEMA - VIA BENEDETTO MARCELLO 1a-1b
• ELETTRONICA CENTOSTELLE - VIA CENTO STELLE 5/a-b
• HELP COMPUTER - VIA DEGLI ARTISTI 15-A
• TELEINFORMATICA TOSCANA -VIA BRONZINO 36
Provincia di Firenze
• WAR GAMES - VIA R. SANZIO 126/A - EMPOLI
• NEW EVM COMPUTER - VIA DEGLI INNOCENTI 2 - FIGLINE VALDARNO
• C.tro INFOR. - VIA ZNOJMO 41 - PONTASSIEVE
• COSCI F.LLI - VIA ROMA 26 - PRATO
• BARBAGLI C. ELET. - VIA F. BONI 80 - PRATO
Grosseto
• COMPUTER SERVICE - VIA DELL'UNIONE 7
Livorno
• ETA BETA - VIA SAN FRANCESCO 30
• FUTURA 2 - VIA CAMBINI 19
Provincia di Livorno
• PUNTO ROSSO - VIA BARONTINI 28 - PIOMBINO
Provincia di Lucca
• IL COMPUTER - V.LE COLOMBO 216 - LIDO DI CAMAIORE
• SANTI VITTORIO - VIA ROMA 23 - S. ROMANO GARFAGNANA
• TOP GAMES - VIA S. ANDREA 122 - VIAREGGIO
Massa
• EURO COMPUTER - P.ZZA G. BERTAGNINI 4
Carrara
• RADIO LUCONI - VIA ROMA 24/B
Pisa
• ELECTRONIC SERVICE - VIA DELLA VECCHIA TRANVIA 10
• PUCCINI S.- CP 1199 (RAG.SOC. MAREX) - VIA C.CAMMEO 64
• TONY HI-FI - VIA CARDUCCI
Provincia di Pisa
• M.C. INFORMATICA - VIA DEL CHIESINO 4 - PONTEDERA (PI)
Pistoia
• ELECTRONIC SHOP - VIA DEGLI SCALZI 3
Provincia di Pistoia
• ZANNI &C. - C.SO ROMA 45 - MONTECATINI T.

Siena
• R. BROGI - P.ZZA GRAMSCI 28
• VIDEO MOVIE - VIA GARIBALDI 17
Provincia di Siena
• ELETTRONICA di BIFOLCHI - VIA DI GRACCIANO NEL CORSO 111 - MONTEPULCIANO

LAZIO
• CENTRO INF. - D.R.R. srl - TEL. 06-5565672

UMBRIA
Perugia
• MIGLIORATI - VIA S. ERCOLANO 3-10
Provincia di Perugia
• COMPUTER STUDIO'S - VIA IV NOVEMBRE 18/A - BASTIA UMBRA
• WARE - VIA DEI CASCERI 31 - CITTA'DI CASTELLO
Terni
• CGS SOFTWARE HOUSE - VIA DONIZETTI 71/A

BASILICATA
Matera
• G. GAUDIANO ELECTRONICS - VIA ROMA ang. XX SETTEMBRE 1

PUGLIA
Bari
• ARTEL - VIA GUIDO D'ORSO 9
• COMPUTER'S ARTS - V.LE MEUCCI 12/B
• PAULICELLI S. & F. - VIA FANELLI 231/C
Provincia di Bari
• F. FAGGELLA - C.SO GARIBALDI 15 - BARLETTA
• G.FAGGELLA - P.ZZA D'ARAGONA 62A - BARLETTA
• LONUZZO G. - VIA NIZZA 21 - CASTELLANA
• TECNOUFF. - VIA RICASOLI 54 - MONOPOLI
• TANGORRA N. - C.SO V.EMANUELE 130/B - TRIGGIANO
Brindisi
• MARANGI E NICCOLI - VIA PROV. SAN VITO 165
Provincia di Brindisi
• MILONE G. - VIA S.F. D'ASSISI 219 - FRANCAVILLA FONTANA
Foggia
• BOTTICELLI G. - VIA SAV POLLICE 2
• E.C.I. COMPUTER - VIA ISONZO 28
• LA TORRE - V.LE MICHELANGELO 185
Provincia di Foggia
• IL DISCOBOLO - VIA T. SOLIS 15 - SAN SEVERO
Lecce
• BIT - VIA 95 REGG.NTO FANTERIA 87/89
Provincia di Lecce
• TECNO UFFICIO - P.ZZA GIOVANNI XXIII 10 - GALLIPOLI
• CEDOK INFORMATICA - VIA UMBERTO I 116 - TRICASE
Taranto
• ELETTROJOLLY C.tro - VIA DE CESARE 13
• TEA - TEC. ELET. AV. - VIA R. ELENA 101

CAMPANIA
Provincia di Avellino
• FLIP FLOP - VIA APPIA 68 - ATRIPALDA
Benevento
• E.CO. INF. - VIA PEPICELLI 21/25
Caserta
• ENTRY POINT - VIA COLOMBO 31
• O.P.C. - VIA G. M. BOSCO 24
Provincia di Caserta
• M.P. COMPUTER - VIA NAPOLI 30 - MADDALONI
• DAMIANO - C.SO V. EMANUELE 23 -ORTA DI ATELLA
• FUSCO B. - VIA NAPOLI 24 - VAIRANO PATERNORA (FRAZ. VAIRANO SCALO)
• LINEA CONTABILE - VIA OSPEDALE 72/76 - SESSA A. (CE)
Napoli
• BABY TOYS - VIA CISTERNA DELL'OLIO 5/BIS
• CASA MUSICALE RUGGIERO - P.ZZA GARIBALDI 74 (INT. STAZ. F.F. S.S.)
• C.tro ELET. CAMPANO - VIA EPOMEO 121
• CI.AN - GALLERIA VANVITELLI 32
• CINE NAPOLI - VIA S. LUCIA 93/95
• DARVIN - CALATA SAN MARCO 26
• GIANCAR 2 - P.ZZA GARIBALDI 37
• ODORINO - L.GO LALA 22 A-B
• R 2 - VIA F. CILEA 285
• SAGMAR - VIA S. LUCIA 140
• TOP VIDEO - TOP COMPUTER - VIA S. ANNA DEI LOMBARDI 12
• VIDEOFOTOMARKET - VIA S. BRIGIDA 19
Provincia di Napoli
• ELECTRONIC DAY - VIA DELLE PUGLIE 17 - CASORIA
• TUFANO - S.S. SANNITICA 87 KM 7 - CASORIA
• SOF SUD - V.LE EUROPA 59 - CASTEL/MARE DI STABIA
• ELETTRONICA 2000 - C.SO DURANTE 40 - FRATTAMAGGIORE
• SPADARO - VIA ROMANI 93 - MADONNA DELL'ARCO
• GATEWAY - VIA NAPOLI 68 - MUGNANO
• VISPINI & DI VUOLO - VIA A.ROSSI 4 - POMPEI
• SPY CASH & CARRY - P.ZZA ARENELLA 6/A - NAPOLI
• NUOVA INFORMATICA SHOP - VIA LIBERTA' 185/191 - PORTICI
• BASIC COMPUTER - C.SO GARIBALDI 34 - POZZUOLI
• V.C. - C.SO SECONDIGLIANO 562/B - SECONDIGLIANO
• F. ELETTRONICA - VIA SARNO 102 - STRIANO
• TECNO - VIA V. VENETO 48 - TORRE DEL GRECO
Salerno
• COMPUMARKET - VIA BELVEDERE 35
• COMPUTER MARKET - C.SO VITTORIO EMANUELE 23
Provincia di Salerno
• KING COMPUTER - VIA OLEVANO 56 - BATTIPAGLIA
• DIMER POINT - V.LE AMENDOLA 36 - EBOLI
• IACUZIO F. - VIA MUNICIPIO 14 - MERCATO SAN SEVERINO
• COMPUTER SERVICE - VIA L.DA VINCI 81 - SCAFATI

CALABRIA
Catanzaro
• C. & G. COMPUTER - VIA F. ACRI 28
• PAONE S. & F. - VIA F. ACRI 93/99
Provincia di Catanzaro
• COMPUTER HOUSE - VIA BOLOGNA (L.GO OSPEDALE) - CROTONE
• RIOLO F.LLI - VIA VENEZIA 1/7 - CROTONE
• ING. FUSTO S. - C.SO NICOTERA 99 - LAMEZIA TERME
Cosenza
• MAISON DE L'INFORMATIQUE - VIA PASQUALE ROSSI 34/C
• SIRANGELO COMP. - VIA N. PARISIO 25
Provincia di Cosenza
• HI-FI ALFANO G. - VIA BALDACCHINI 109 - AMANTIA
• ELIGIO ANNICCHIARICO &C. - VIA ROMA 21 - CASTROVILLARI
• ALFA COMPUTER - VIA NAZIONALE 341/A - CORIGLIANO SCALO
REGGIO CALABRIA
• CONTROL SYSTEM - VIA S.F DA PAOLA 49 D
• SYSTEM HOU. - VIA FIUME ang. PALESTINO 1
Provincia di Reggio Calabria
• COMPUTER SHOP - V.LE MATTEOTTI 36/38 - LOCRI
• PICIEFFE - C.SO F. S. ALESSIO 19 - TAURIANOVA
SICILIA
• CENTRO INF. - ITALSOFT SRL - TEL. 0935-696090

Leggo VR perché mi dà la rotta

IMMAGINE MARSILII